PRIMERA REVISTA DE PROGRAMACIÓN EN CASTELLANO

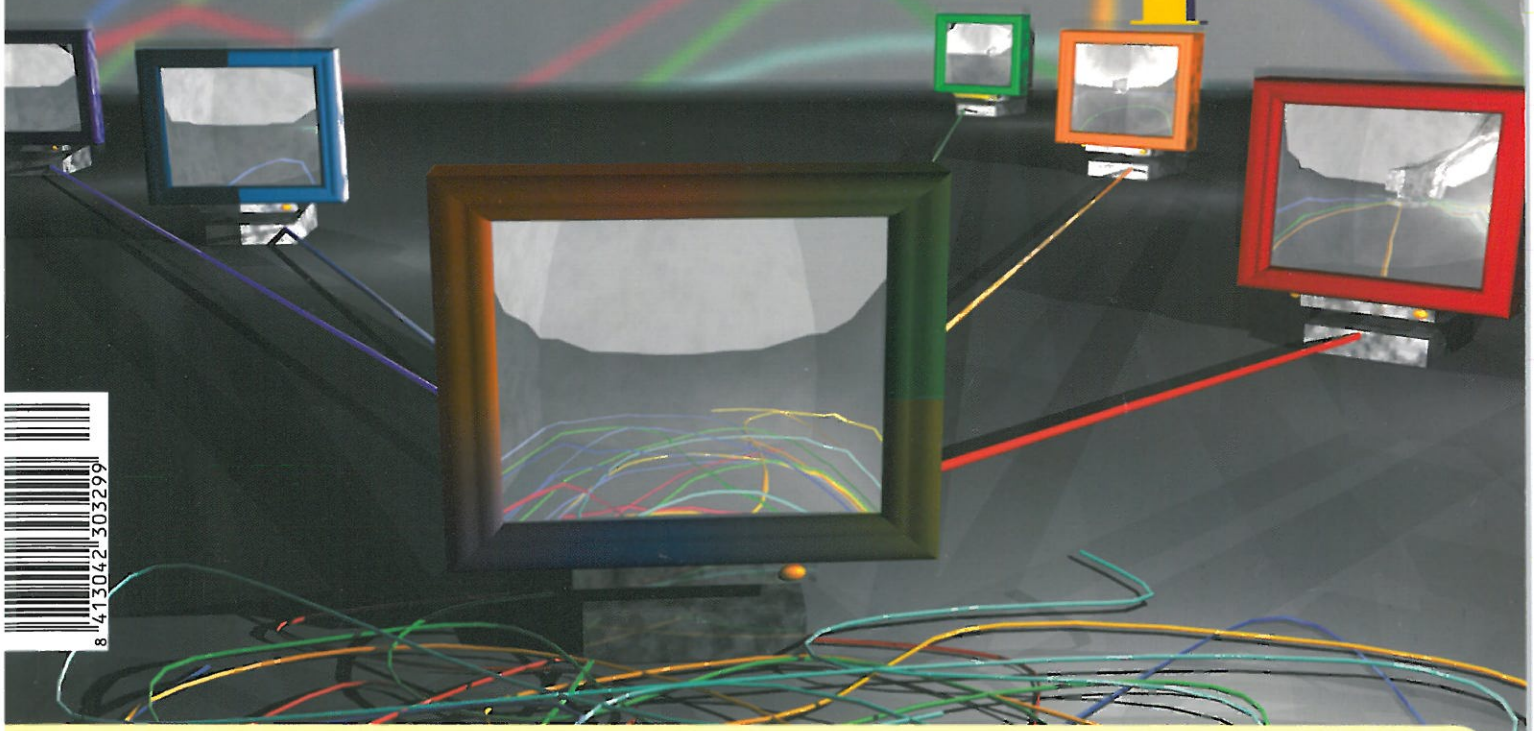
SOLO PROGRAMADORES

AÑO VI. 2.ª ÉPOCA • NÚMERO 70 • UNA PUBLICACIÓN DE: REVISTAS PROFESIONALES S.L. • Precio: 995 Ptas. 5,98 € (IVA incluido)

CD
URSOS DE
VB 6
HTML &
vaScripts
85
LICACIONES
LIBRERÍAS

Windows 2000 ADS:

Para ver los árboles del bosque



8 413042 303299

WINDOWS: Scripting Host
PROGRAMACIÓN WEB: XML & Java
TELECOMUNICACIONES: WAP

DELPHI: Texturas OpenGl
BBDD: Aplicaciones con Delphi 5
LINUX: Páginas dinámicas PHP

JAVA : Java Server Pages
Optimación de aplicaciones
DUDAS: Consultorio técnico

© 2000 IBM Corp.

SUMARIO 70

SÓLO PROGRAMADORES

5 CONTENIDO DEL CD-ROM
Cursos completos de *VB6* y *HTMl-JavaScript*, además de un montón de ejemplos, códigos fuente, actualizaciones, herramientas...

6 NOTICIAS
En agosto también pasan cosas, y las noticias no se van de vacaciones. Los problemas de *Microsoft* con la *UE*, los resultados de la campaña antivirus del Gobierno, o las hazañas de un progamador de 17 años son algunas de las noticias que hemos seleccionado.

8 PORTADA WINDOWS - WINDOWS 2000 ADS
Descubre uno de los aspectos más interesantes del *Windows 2000*. El DA es una excelente herramienta para la administración de redes corporativas.

16 WINDOWS. WSH, ¿HERRAMIENTA O AMENAZA?
Windows Scripting Host es una poderosa herramienta en manos del progamador experto, sin embargo, su mal uso supone una auténtica amenaza.

22 PROGRAMACIÓN WEB. XML&JAVA (II)
Continuamos con esta serie dedicada a este potente *tandem* para el desarrollo de aplicaciones *web*. Continuamos con la estructura interna de un documento *XML*.

28 JAVA. PÁGINAS JSP (I)
Sun no es ajena al desarrollo de *Internet*. *Java Server Pages* es su propuesta para la creación de aplicaciones *web* versátiles.

34 OPTIMACIÓN DE APLICACIONES JAVA
¿Puede mejorarse el rendimiento aplicaciones *Java* sin necesidad de recurrir a programas específicos? Aquí tienes la respuesta.

40 TELECOMUNICACIONES. CURSO DE PROGRAMACIÓN CON WAP (V)
Un artículo más de esta interesante serie que nos descubre cómo llevar los contenidos de *Internet* al teléfono móvil.

46 BASES DE DATOS. APLICACIONES CON DELPHI (y VII)
Llegamos al final de la serie con los componentes *IBX* y las actualizaciones en *caché* para mejorar el rendimiento.

51 LINUX. PÁGINAS WEB DINÁMICAS CON PHP (I)
PHP es uno de los lenguajes de *scripts* preferidos para *Linux*, debido a su sintaxis simple y facilidad para interactuar con *BBDD* diferentes.

53 DELPHI. PROGRAMACIÓN OPEN GL EN DELPHI (V)
En este número, aprenderemos a usar las texturas para mejorar el aspecto gráfico sin generar un exceso de polígonos.

63 LIBROS
Seguro que encontráis interesante esta sección de novedades editoriales: *Dreamweaver 3* y *Html 4*.

64 DUDAS TÉCNICAS
¿Sabes cómo hacer funcionar un programa que muestre la memoria disponible en tu equipo en tiempo real? Si buscas soluciones a cualquier problema técnico, esta es tu página.

SÓLO PROGRAMADORES

Número 70
SÓLO PROGRAMADORES
solop@revistasprofesionales.com
es una publicación de
REVISTAS PROFESIONALES S.L.
http://www.revistasprofesionales.com

Editor
Agustín G. Buelta
Coordinador Técnico
Eduardo De Riquer Frutos
Coordinador de Redacción
Francisco Pino García
Edición
Gema Prados Mateos
Colaboradores
Constantino Sánchez, Juan Luis Ceada, Javier Sanz, Adolfo Aladro, Enrique de la Lastra, Marc Vaquer Jordi Agost, Vicente A. Sánchez Werner
Imagen de portada
José Antonio Cagigal
Maquetación y Tratamiento de Imagen
Paco Risco
Consultas técnicas
atecnica@virtualsw.es

Asesoría de Publicidad
Carmina Ferrer
Tel.: (91) 304 78 46
Mariano Sánchez (Barcelona)
Tel.: (93) 322 12 38

Suscripciones
Rosa Tabares
Tel. (91) 304 87 64 Fax: (91) 327 13 03

Impresión
Printerman
Distribución
Motorpress Ibérica
GRUPO MPIB Motorpress-Ibérica

La revista Sólo Programadores no tiene por qué estar de acuerdo con las opiniones escritas por sus colaboradores en los artículos firmados. El editor prohibe expresamente la reproducción total o parcial de los contenidos de la revista sin su autorización escrita.

Depósito legal: M-26827-1994
ISSN: 1134-4792
PRINTED IN SPAIN
COPYRIGHT 31-10-2000
Precio en Canarias, Ceuta y Melilla:
1.202 ptas. sin I.V.A.
Con sobretasa aérea: 1.275 ptas.

AEPP
Asociación Española de Editoriales de Publicaciones Periódicas

EDITORIAL

Lo prometido es deuda

En el número anterior, os anunciábamos cambios en la revista, y aquí tenéis los primeros resultados. En el tórrido mes de julio -Dios mío, ¡qué calor!- nos pusimos manos a la obra para poder ofreceros aquella revista más participativa y actual que os prometimos, y aunque todavía quedan algunos retoques, así nos ha salido la criatura: Más contenido. El nuevo modelo de página nos ha permitido aprovechar más espacio, de manera que podemos ofreceros más artículos sin mermar la oportunidad y calidad con las que estamos comprometidos. Esto nos va a dejar también acortar las series, de forma que podáis disponer de los cursos completos más rápidamente.

Suplemento de *Internet*. No podemos ser indiferentes al creciente protagonismo cobrado por *Internet* en todos los ámbitos de la sociedad, y este protagonismo no es ajeno al desarrollo de las últimas tecnologías de programación informática. Desde hace meses, los artículos relacionados con *La Red* han ido aumentando en número e interés en nuestra revista, y se han hecho acreedores de su propia sección fija. A partir de ahora, la podréis encontrar en una separata, para que os sea más fácil clasificarla y buscar los artículos. En este *número 0* hemos querido presentaros a C#, el nuevo lenguaje de *Microsoft* que viene a plantar cara al omnipresente *Java*, y que sin duda, animará el panorama de la programación *web*.

Un artículo especialmente interesante es el dedicado al *Directorio Activo de Windows 2000*, que nos descubre una de las mejores virtudes del nuevo sistema operativo sucesor de *NT*. Ah, y no paséis por alto el artículo dedicado a *Java Server Pages*, pues ante el envite de *Microsoft, Sun* no se duerme en los laureles...

Ha sido un placer ver cómo aumentaban vuestras sugerencias y consultas recibidas en la redacción, y os aseguramos que tomamos buena nota de todas ellas, para ir completando poco a poco la mejor revista de programación de habla hispana. Como hacíamos antes, os animamos encarecidamente a que continuéis mandándonos vuestras opiniones, críticas y sugerencias.

La noticia que no nos hubiera gustado dar, es que el esfuerzo que hemos hecho para mejorar la revista, ha supuesto un aumento de los costes, que nos vemos obligados a repercutir -sólo en parte- en el precio final de la publicación. Confiamos en compensaros sobradamente de esas 20 pesetas más...

ENTORNOS DE PROGRAMACIÓN

)VENTURE MAKER 2.05
reación de juegos interactivos de 'enturas.
:V-C++ 3.99
ntorno de desarrollo para programas 'C++.
EE PASCAL 1.00
ompilador *freeware* de *Pascal* de 32 *ts*.
:LP MAGICIAN PRO 4.60
erramienta para generación de fiche- s de ayuda de *Windows*.
VENSTEPS 2.1
tilidad de creación de sistemas profe- onales de ayuda en línea.

LENGUAJES

32 ACTIVEX CONTROL 1.1.0
ene un sistema de actualización para adir o eliminar campañas publicitarias.
FASTFORM 3.6.47
ontrola el funcionamiento de formula- os.
30CX 1.01
ontrol *VB* de creación de reproduc- res *MP3*.
ANTOMDIRECT 2.0.7.0 BETA2
erramientas *VB* para codificar aplica- ones dinámicas.
/A WEB START API 0.4
icia aplicaciones desde el navegador.

JBUILDER FOUNDATION 3.5
Construcción de aplicaciones *Java, applets, JSP/Servlets, JavaBeans, EJB* y *CORBA*.
UNIFY EWAVE SERVLETEXEC 3.0
Servidor de aplicaciones *Web* y motor *servlet* para componentes de *Java Servlet API* y *JavaServer Pages (JSP)*.
DHTML MENU BUILDER 2.4
Creación de menús *DHTML*.
WAPTOR 2.3
Sencillo y potente editor *WML*.
WEBBERACTIVE 4.0
Editor para crear documentos en *HTML* dinámico.
FLIPPER CAD CONTROL 2.5
Control *ActiveX* para incluir funciones de *CAD* a un programa.
MILESTONE 16
Avance de la nueva tecnología *Internet* de *Netscape*. Versión para expertos.
VIRTUAL WINDOWS CLASS LIBRARY (VWCL)
Completa colección de clases *C++*.

HERRAMIENTAS

ALFA INTERCEPTER 6.0
Componente de protección de ficheros.
LEMMY 4.1
Versión 32 *bits* del editor *UNIX vi*.
PORTMON 3.02
Monitoriza los puertos serie y paralelos de un sistema.
PROCESS CONTROL TASK MANAGER 2.0
Gestor de tareas que muestra todos los procesos en ejecución.
VBTRANSFORM 1.01
Convierte formularios *FRM* de *VB*en código *C++*.
VISUAL BUILD 2.0
Herramienta de gestión que automatiza tareas.
Z-UP MAKER 2.2
Creación de programas de instalación.

REDES

COMMUNICATE PRO 2.0.30
Centro de comunicaciones con listas de teléfonos, fax, buscapersonas, *Internet*, etc.
INTRANETMANAGER 1.2
Conecta *intranets* a *Internet* sin *gateway*.
IPCHECK 2.3
Análisis de rendimiemto de servidores.
JRUN SERVER 3.0
Servidor de aplicaciones *J2ee* y entorno de desarrollo integrado.
SUMMARY PRO 1.4.7
Seguimiento del servidor *Web*. Con más de 60 informes.
WEBSTRESS 3.0
Analiza el rendimiento del servidor *Web*.

DOCUMENTACION

CRASH COURSE DELPHI 1.0
Tutorial de *Delphi* para principiantes.
JAVA 2 DEVELOPMENT KIT DOCUMENTATION 1.3
Documentación del *kit* de desarrollo de *Sun JDK*.
LEARN VISUAL BASIC 6 V2.0
Completo manual de *VB 6*, con 60 ejemplos.
WEB CONCEPTS 2000 V2.0
Curso de *HTML* y *JavaScript*.

IMPRESCINDIBLES

ANTIVIRUS
AntiViral Toolkit Pro 3.0.129, McAfee VirusScan 5.0.1, Panda Antivirus 6.19.00 Platinum.

GRÁFICOS
Icon Bank 4.0, IconForge 4.7, MicroAngelo 98 v4.77, Paint Shop Pro with Animation Shop 6.02, Reptile 2.01, SureThing CD Labeler 2.10.15, ThumbsPlus 4.10-S, Ultra Fractal 2.04, Xara WebStyle 1.2, Xara 3D 4.0.

INTERNET
Copernic 2000 v4.55 SP, Cuentapasos 3.79, CuteFTP 4.0, Dial-Up Magic 1.8, Eudora Pro 4.3.2, GetRight 4.2c, Go!Zilla 3.5, Guardian 1.1, i.Share 3.5, Jet Color 1.20, Mass Downloader 1.2.95-SR2, MIRC 32 5.71,SubmitWolf ES 4.02.002, XMLwriter 1.21.

MULTIMEDIA
AudioCatalyst 2.01, COWON Jet-Audio 4.7, Sonique 1.51, WinAMP 2.64.

NAVEGADORES
Explorer 5.5, Netscape Communicator 4.73, Opera 4.01.

PROGRAMACION
Decafe Pro 3.6, Hackman 4.04, UltraEdit Prof. 7.10, W Registry Guide 1.3, WinHex 9.45.

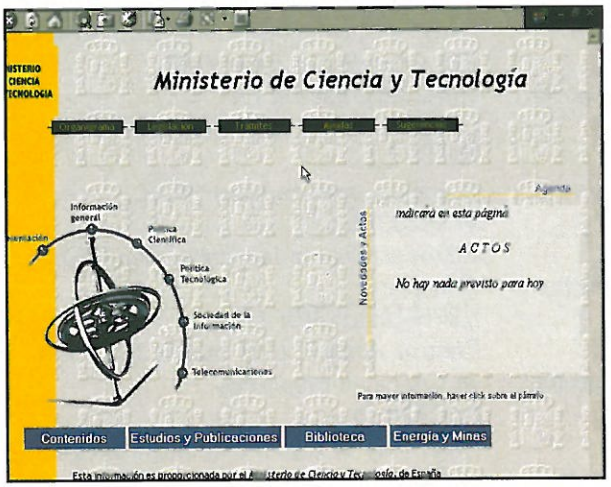
UTILIDADES
Adobe AR 4.05, DirectX 7.0a, Emergency Recovery System 9.05, Nero 5.0.0.9, Windows Commander 4.50, WinZip 8.0.

ATENCIÓN

En caso de tener algún problema con el CD-ROM, envíelo por correo ordinario a la atención del SERVIDIO TÉCNICO de SÓLO PROGRAMADORES, incluyendo en el interior del sobre sus datos personales, a la siguiente dirección:
San Sotero, 5 - 1.º planta
28037 MADRID

CRÉDITOS SIN INTERÉS PARA CONTENIDOS EN ESPAÑOL EN LA RED

El *Ministerio de Ciencia y Tecnología* ha dotado con 53.000 millones de pesetas una partida especial del *Plan Nacional de I+D+i*, destinada a fomentar los contenidos en español dentro de *Internet* por medio de créditos de interés 0, concedidos a las empresas que presenten proyectos dirigidos a las tecnologías de la información, comercio electrónico y creación de contenidos digitales que promocionen la presencia del mundo hispanohablante en *La Red*. Esta financiación consistirá en créditos reembolsables, al 0 % de interés, reintegrables en un plazo de siete años y dos años de carencia. Sin embargo, la oferta estará limitada a aquellas compañías que además del interés del proyecto, aporten un aval, para que "esta ayuda se configure, como un "instrumento adecuado para empresas consolidadas que quieren abrir una nueva línea de investigación, o para las que cuenten con algún tipo de valedor que pueda aportar por ellas dicho aval, en una especie de padrinazgo empresarial".



UN RELOJ CON CORREO ELECTRÓNICO

Los avances de la tecnología *WAP* en el ámbito de la telefonía móvil, han abierto caminos insospechados hace tan sólo unos meses. Puestos a rizar el rizo, La multinacional *IBM* está desarrollando un prototipo de reloj de pulsera gobernado por *Linux*, que permitirá conectarse sin hilos al *PC*, teléfonos móviles y otros aparatos, para recibir mensajes de correo electrónico y *SMS*. *IBM* pretende matar dos pájaros de un tiro, al adelantarse en la posibilidad de ofrecer contenidos *WWW* en dispositivos minúsculos, y demostrar la idoneidad de *Linux* para este tipo de accesorios. El reloj-mensajero mide cinco centímetros de largo por uno de ancho, e incorpora un versátil procesador y una memoria flash de 8 *megabytes*, aunque no está disponible por el momento en el mercado.

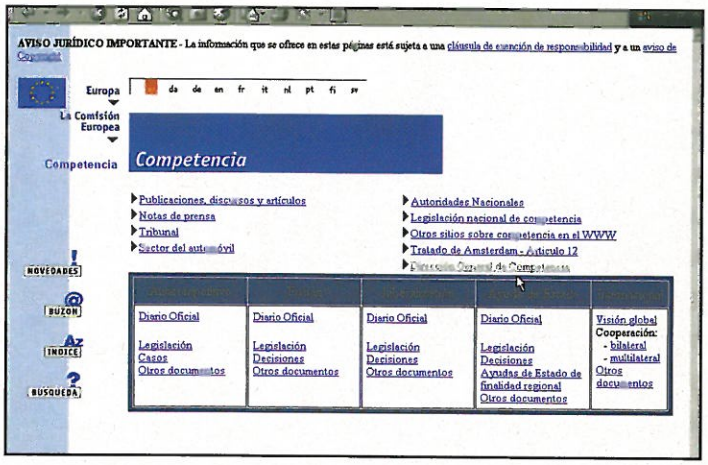
UN PROGRAMADOR DE 17 AÑOS CREA UN INTERCAMBIADOR DE JUEGOS

Jeff Phillips Freeman es un joven (17 añitos) progamador informático de Estados Unidos, que ha alcanzado en los últimos días una extraordinaria popularidad entre sus compatriotas. La razón no es otra que haber sido el creador de *Swapoo*, un *software* destinado al intercambio de videojuegos de forma gratuita, como si de cromos se tratara, al estilo de lo hecho por *Napster* en el terreno musical. Esta feliz ocurrencia del precoz progamador, no parece haber hecho mucha gracia a compañías como *Nintendo*, o *Sega*, creadores de los juegos *Pokémon*, y *Tetris*, quienes se han apresurado a denunciar lo que para ellos no es más que un nuevo tipo de piratería. El caso, es que desde la aparición de la página http://swapoo.zophar.net a principios de agosto, se han registrado más de 8.000 visitas diarias.

El joven Freeman se ha mostrado sorprendido por tanto revuelo (su historia se publicaba en *USA Today*), al tiempo que preocupado por los problemas que la ley de propiedad intelectual y *copyright* le puedan deparar, pues las grandes multinacionales no se andan con juegos, y se toman muy en serio este tipo de piratería.

LA COMISIÓN EUROPEA DE LA COMPETICIA, A POR MICROSOFT

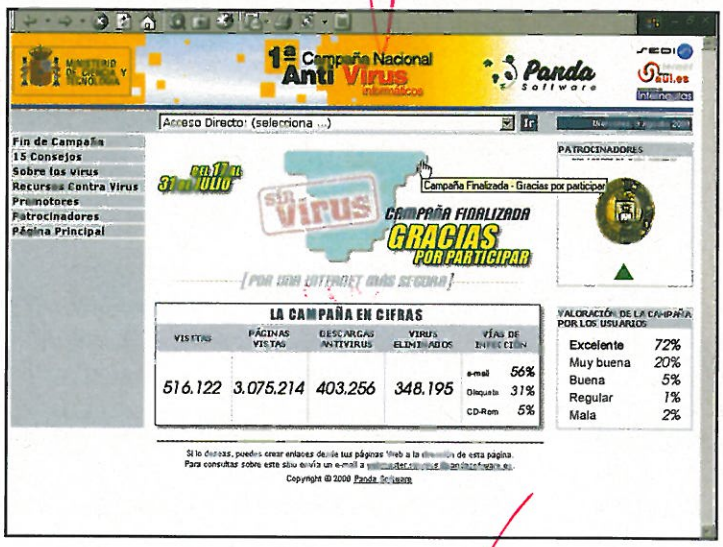
Los problemas jurídico-comerciales de *Microsoft* aumentan. Ya no se trata sólo del contencioso mantenido entre la justicia estadounidense y el imperio de *Mr. Gates*...a primeros de agosto, la *UE* aceptaba la denuncia presentada por *Sun Microsystems* en diciembre de 1998 que acusaba a su rival de prácticas monopolísticas en el mercado de los sistemas para ordenadores personales, "imponiendo la obligación de dar acceso a sus interfaces y asegurar la interoperabilidad con los programas de otros fabricantes". *Microsoft* dispone de dos meses, a partir de la admisión del procedimiento, para alegar y responder a las denuncias de *Sun* ante la *Comisión de la Competencia*. Bruselas encuentra indicios de que *Microsoft* puede haber abusado de su posición dominante en los sistemas operativos de ordenadores personales con el *Windows 2000*. La *UE* considera que *Microsoft* -quien copa el 95% del mercado de sistemas operativos- sólo ha dado informaciones fragmentadas, voluntariamente incompletas, y con una intención discriminatoria hacia sus competidores. Mario Monti, *Comisario Europeo* de *Competencia* ha manifestado que "la *Comisión* no



tolerará que una posición dominante en un mercado concreto se extienda a mercados próximos basándose en prácticas contra la competencia con el pretexto de proteger sus derechos de autor".

RESULTADOS DE LA CAMPAÑA ANTIVIRUS DEL GOBIERNO

La campaña organizada por el *Ministerio de Ciencia y Tecnología* y la compañía *Panda Software* para la prevención y reparación de daños causados por virus informáticos ha terminado en su primera fase con un total de 500.000 visitas a *sinvirus.com*, la página web habilitada para información y desinfecciones de emergencia, en la que durante la última quincena de julio, *Panda* suministró gratuitamente más de 400.000 copias del antivirus *Panda Antivirus Lite*, rea-

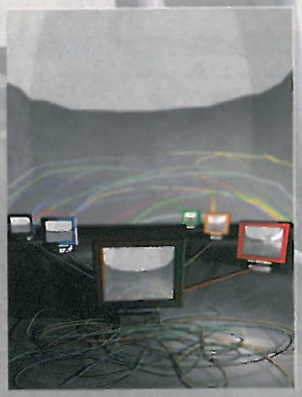


lizándose un total de 350.000 desinfecciones. La información más demandada ha sido la relacionada con los gusanos *Pretty-Park* y *Happy99*, que afectan al correo electrónico y acceden a la información privada.

Además, de *Panda Software*, también colaboraron en la campaña la *Asociación de Usuarios de Internet* y la *Asociación de Internautas*, quienes han manifestado su intención de extender estas iniciativas a los servicios prestados por navegadores como *Netscape*.

PARCHES PARA WINDOWS 2000

La primera entrega de programas correctores de *Windows 2000* ya está disponible en la página de la compañía, desde donde se puede descargar o solicitar su *CD*. El *Service Pack 1*, de 33 *MB*, viene a corregir los últimos problemas de estabilidad detectados, con lo que *Microsoft* quiere presentar a su sistema operativo como el más fiable del mercado, aun por encima de *Linux*. Hasta el momento, *Microsoft* asegura haber vendido 4 millones de copias de *Windows 2000*, y vaticina que este programa terminará por ser usado por el 60% de las grandes y medias compañías de todo el mundo.



Windows 2000 ADS

Para ver los árboles del bosque

Marc Vaquer Crusat. *Técnico de Sistemas.*

Microsoft aborda definitivamente los servicios de directorio con _Windows 2000_. Su nuevo sistema operativo permite administrar una gran red corporativa.

Para diseñar y mantener un *Directorio Activo* (*Active Directory*, o *AD*) en *Windows 2000* para una empresa, primero hay que tener claros todos los conceptos básicos. Aunque hay libros enteros dedicados a esto, aquí resumiremos lo más necesario e importante. Empezaremos por los términos: es difícil exagerar la importancia de entender la terminología y funcionamiento del *Directorio Activo* antes de iniciar un diseño.

Comprender la terminología es fundamental

Con el *Directorio Activo*, toda la información sobre usuarios, ordenadores, impresoras, aplicaciones y ficheros, está disponible para su uso en la red local. La estructura de la información es como la estructura de tu organización, facilitando la búsqueda de datos, como impresoras, o el *e-mail* de un compañero.

CARACTERÍSTICAS MÁS IMPORTANTES

El *AD* amplía el modelo de *Dominio* del *Windows NT 4* haciéndolo más eficiente, y añade muchas nuevas características:

- Modelo de *Dominio*. Los *Dominios* en *NT* eran estructuras planas limitadas a 40.000 objetos, con consecuencias poco afortunadas. Por un lado, la asignación de permisos era del tipo *todo o nada*. Sin herencia dentro del *Dominio*, sus limitaciones obligaban a tener varios *Dominios* que podían llevar a una red inadministrable. Con el *Directorio Activo*, los *Dominios* son jerárquicos, y virtualmente sin limitaciones. Ahora, los administradores pueden delegar autoridad en un grupo pequeño de *Dominios*.
- *Relaciones de confianza* transitivas (*Trusts*). En *NT 4*, gestionar las *relaciones de confianza*, unidireccional y manualmente, podía ser un lío. Había

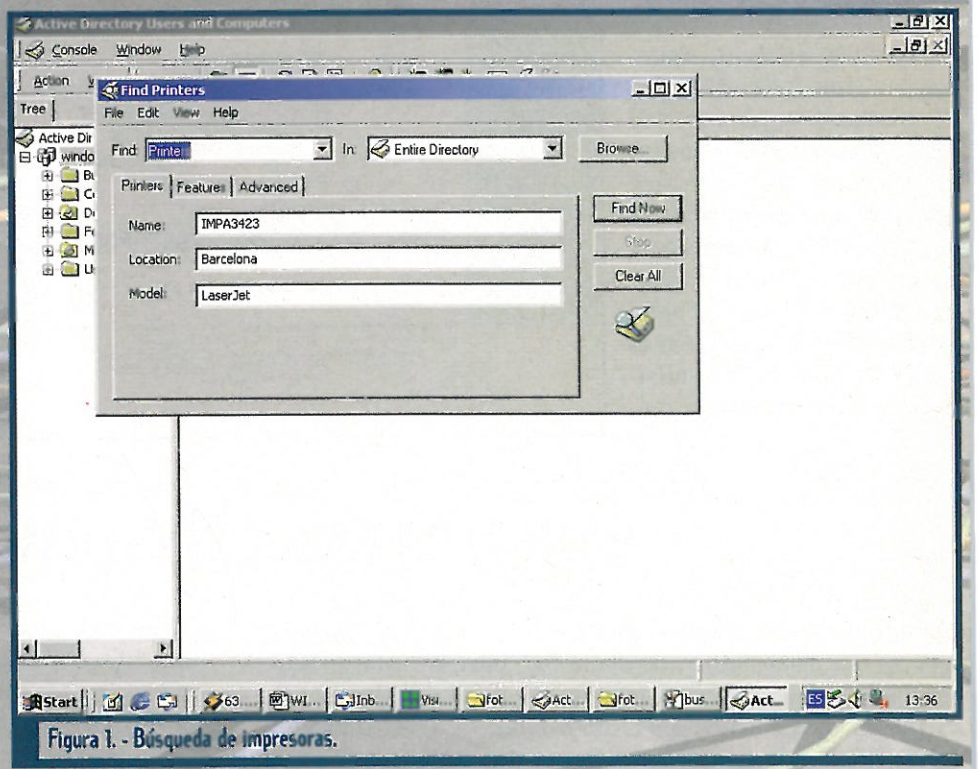


Figura 1. - Búsqueda de impresoras.

que especificarlas individualmente, y si el *Dominio A* confiaba en *B*, y *B* en *C*, *A* no confiaba en *C*, si no indicábamos lo contrario. El *AD* incluye *relaciones de confianza* bidireccionales y transitividad automática.

- *Políticas de grupo*. Permite especificar perfiles con información de configuración en la jerarquía de *Dominio*. Al añadir usuarios les puedes configurar o añadir programas automáticamente.
- *Réplica Multimaster*. Cada controlador de *Dominio* propaga automáticamente sus objetos a los demás, así que todos tienen toda la información del *Dominio*, y si falla alguno, el *AD* sigue siendo accesible. Ya no hay controladores primarios y de Backup.
- *Catálogo Global*. Búsqueda eficiente: todos los objetos y sus atributos más frecuentes se guardan en una réplica parcial de cada *Dominio Windows 2000* en un directorio, el *Catálogo Global*, construido automáticamente durante la replicación del *AD*
- *Cumplimiento de estándares*. El *AD* está basado en el *Lightweight Directory Access Protocol* (*LDAP*) y otros estándares. Es posible integrarlo con otros servicios de directorio, y añadir componentes de terceros.

GUARDAR OBJETOS EN EL AD

os objetos se crean en una estructura jerárquica. Las estructuras se mon-an con dos tipos de objetos: contenedo-es, y hojas. De los contenedores pue-en colgar hojas u otros contenedores.)e las hojas no cuelga nada. Los conte-edores cuelgan a su vez del contene-or raíz, el *"root"*. Los contenedores ıás comunes son la *Unidad* de *Organi-ıción (OU)*, y el contenedor genérico, implemente *Contenedor*.

Cada objeto guardado en el *AD*, ha de er identificable y localizable. Por eso, ada objeto tiene su *Globally Unique ientifier* (*GUID*) asignado. *Microsoft* ga-antiza que este número de 128 *bits* es nico. Además del *GUID*, también se uede usar su *AdsPath* para identificar n objeto.

Los caminos de la jerarquía de un D son los *AdsPaths*, y se usan para lentificar un objeto de forma única. Se resentan con la sintaxis y reglas defi-idas en *LDAP*. Un ejemplo de un *"ro-"* sería: *LDAP://dc=ARTIC,dc=com*.

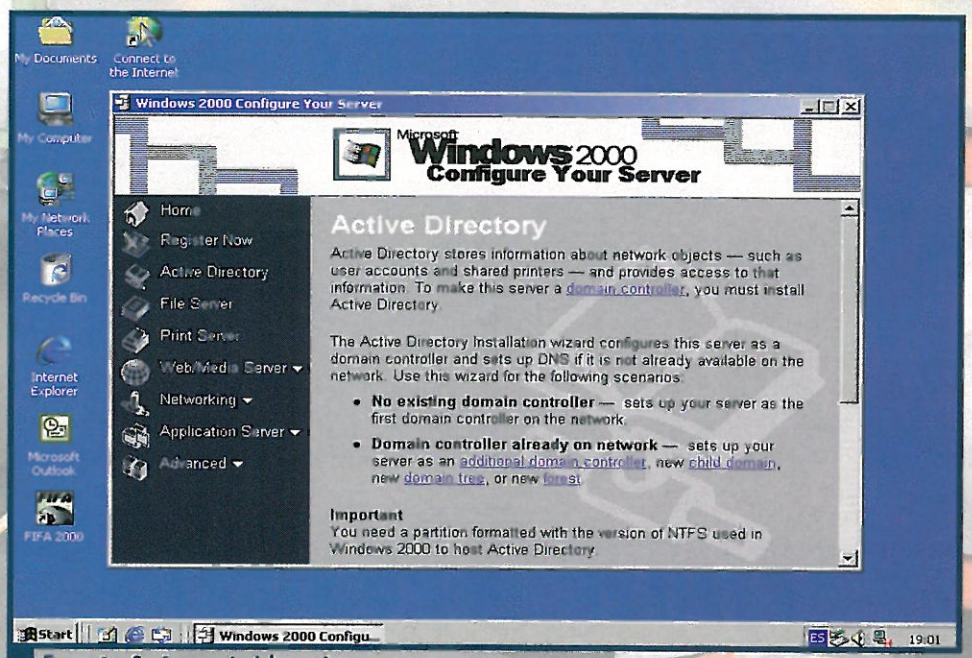


Figura 2. - Configuración del servidor.

Todos los caminos *LDAP* empiezan con un *progID* (identificador de programa) seguido de ":" y doble barra "//". Si el *Dominio* fuera *"comerciales.ARTIC.com"*, el *AdsPaths* sería: *"LDAP://ou=comerciales,dc=ARTIC,dc=com"*. Delante de cada parte se pone el tipo de objeto que es. El *AD* usa sólo tres tipos (aunque *LDAP* tiene muchos más):

- *DC, (Domain Component)*, usado en las raíces. No confundir con *Controlador de Dominio*, también abreviado *DC*.
- *OU* para distinguir *Unidades de Organización*.
- *CN* para *Contenedores* genéricos.

DISTINGUISHED NAMES

Distinguished Name (*DN*) es un nombre usado para referirse de forma única a un objeto en un *Directory Information Tree* (*DIT*), *o Árbol del Directorio de Información*. Un *DN* para un usuario podría ser (notar que es un *AdsPath* sin *ProgID*): *Cn=Marc Vaquer, Ou= Técnicos,dc=ARTIC,dc=com*. *Un Relative Distinguished Name* es la referencia de un objeto dentro su contenedor *padre*, como por ejemplo, *Cn=Marc Vaquer*

DOMINIOS, ÁRBOLES, BOSQUES...

El espacio de nombres del *AD* se estructura en *Dominios*, *Árboles*, *Bosques* y *Unidades Organizativas* (*OU*). La estructura lógica de *Windows 2000* se monta sobre el concepto de *Dominio*. Pero no nos confundamos, no es el mismo *Dominio* que en *NT*, pues se basa en:

- Una estructura jerárquica *X.500* de contenedores y objetos.
- Un nombre de *Dominio DNS* como identificador único. Los nombres *DNS* han desplazado a los nombres *Netbios*.

Los objetos se crean en una estructura jerárquica

- Un entorno de seguridad gestionado por un administrador de *Dominio*, que valida los accesos a los recursos por cuentas en el *Dominio* o *confianzas* con otros *Dominios*.
- Una o más políticas que determinan las funciones restringidas para los usuarios o máquinas en ese *Dominio*.

Un *Dominio* es el componente más pequeño reproducible en un simple *Controlador de Dominio* (*DC*). Si una empresa pequeña ha cogido un *Dominio DNS* llamado *ARTIC.com*, ese será su *root*. El *Dominio ARTIC.com*, es automáticamente creado como el nodo raíz de la estructura jerárquica *Árbol de Dominio*. Es una serie de *Dominios* conectados entre sí jerárquicamente, según un esquema de nombres contiguos. Por eso, cuando los departamentos de *Ventas*, *RRHH*, y *Marketing* quieran su propio *Dominio*, los nombres serán: *ventas.ARTIC.com*, *rrhh.ARTIC.com*, y *marketing.ARTIC.com*. Cada *Árbol* se

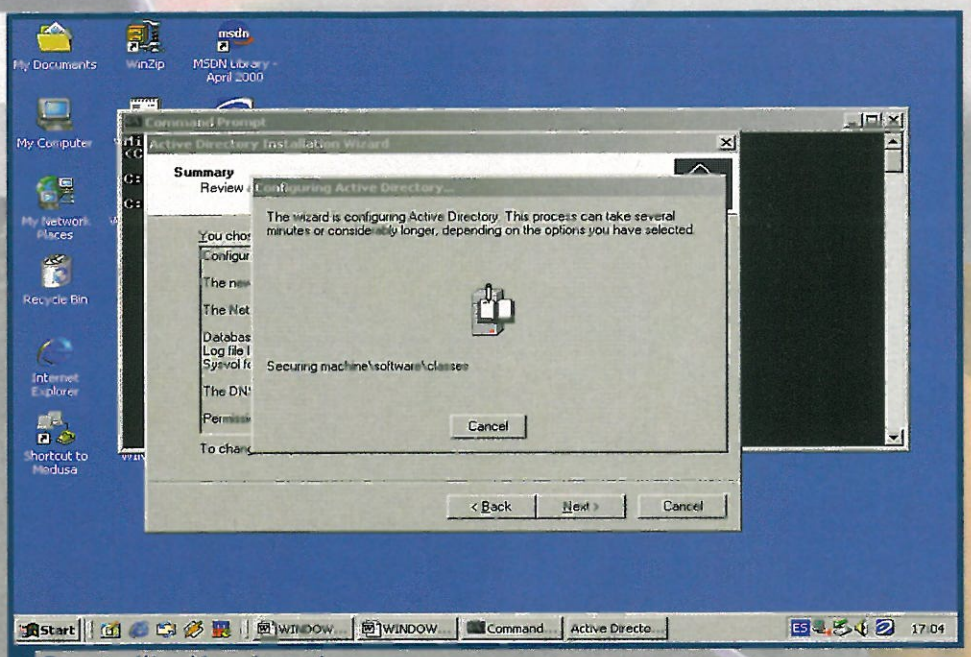


Figura 3. - *Wizard* de configuración.

nomina según el nombre de la raíz del *Árbol*. Es más, este *Árbol* de *Dominio* se llama el *Árbol ARTIC.com*. Los *Dominios* de un *Árbol* confían entre sí implícitamente. El administrador de *ventas.ARTIC.com* puede hacer que cualquier usuario del *Árbol* acceda a los recursos de *ventas.ARTIC.com*. Equivale al modelo de *NT 4.0* de *Confianza Total*. Tantas confianzas entre *dominios*, no afectan a la seguridad, pues aún hay que permitir el acceso a los usuarios de otros *Dominios*.

Los nombres DNS han desplazado a los Netbios

Imaginemos que *ARTIC* tiene una filial llamada *Arrows*. Ésta tendrá un *Dominio arrows.com*. Recordemos que ya tenemos un *Árbol* creado: *ARTIC.com*. De hecho un nuevo *Bosque* se creó con sólo un *Árbol* como miembro: el *Árbol de Dominio ARTIC.com*. Un *Bosque* es un conjunto de *Árboles de Dominio* que confían entre ellos de la misma manera que los *Dominios* de un *Árbol* lo hacen. Las *Confianzas* entre *Árboles* son también transitivas. El nombre de los *Bosques* viene del primer *Árbol* que se creó, el *Dominio* Raíz del *Bosque*, aunque un *Bosque* no tiene estructura jerarquizada. Este primer *Árbol* es importante, pues tiene propiedades especiales, y si destruyes este *Árbol*, destruyes el *Bosque*. Creamos el *Árbol arrows.com* dentro del *Bosque ARTIC.com* (podemos crear un *Bosque* aparte, pero para compartir recursos hay que poner el *Árbol* en el *Bosque* existente). *ARTIC.com* ya tiene dos *Árboles*: *ARTIC.com* y *arrows.com*.

Hemos hablado del *AD* a grandes rasgos: *Dominios*, *Árboles* y *Bosques*. Hablemos ahora de la pequeña escala. Al mirar un *Dominio* de *Directorio Activo*, ves una estructura jerárquica de objetos. Algunos actúan como contenedores y otros no. La *Unidad Organizativa* (*OU*) es el componente más alto para contener objetos. Más abajo, usarás el componente *Contenedor* (*C*), que es un contenedor genérico. La *OU* tiene propiedades de *Dominio*, un *C* no. Ambos pueden contener muchísimos contenedores y objetos, pero la *OU* tiene un entorno de seguridad adicional.

La *OU* será tu principal bloque de construcción dentro un *Dominio*. Si en tu *Dominio* hay pequeños grupos que quieren administración propia, les crearás una *OU* y administrador propio. Se usa un *Contenedor* cuando no hace falta una *OU*. No hay nada que un *Contenedor* pueda que una *OU* no. Naturalmente, cuando se instala *W2000* hay *Contenedores OU* por defecto (*Users*, *Computers*...)

CONSTRUCTORES DEL AD

El *AD* tiene 3 bloques: *Contexto de Nombres de Dominio* (*NC* de *Dominio*), *Contexto de Nombres de Configuración* (*NC* de *Configuración*), y *Contexto de Nombres de Esquema* (*NC* de *Esquema*). Cada uno representa (o almacena) un aspecto del *AD*. Estos bloques se llaman *Particiones* o *Contextos* de *Nombres* (*NCs*).

El *NC* de *Dominio* guarda la información de usuarios, grupos, y *OUs*. Los *NCs* están separados por razones de réplica, pero el *AD* los mantiene unidos, para poder consultarlos con *LDAP*. Un *Bosque* puede tener tantos *NCs* de *Dominio* como quieras, pero sólo un *NC* de Configuración y un *NC* de *Esquema*.

NC Configuración es un *NC* con el nombre *LDAP://cn=Configuration,dc=Dominio-raiz-Bosque, dc=Dominio*... se le llama simplemente "la Configuración", y se replica a cada controlador de *Dominio* del *Bosque*. Guarda la siguiente información:

- Capa física del *site*. Representa todos los sites, subredes, conexiones de replicación y forma de conexión de cada controlador de *Dominio* (*DC*).
- Estructura de los *Árboles* del *Bosque*. Guarda una lista de todos los *Dominios* de cada *Árbol* de *Dominio* y de todos los *Árboles* de cada *Bosque*. También se guardan las *Relaciones de Confianza* manuales entre *Árboles* de diferentes *Bosques*.
- Información global de configuración de los servicios. La configuración de servicios como el *Directorio Activo*, *Exchange Server* y otros de terceros se guardan aquí.

Al *NC* de *Esquema* se accede independientemente con *LDAP://cn=Schema, cn=Configuration, dc=Dominio-raiz-Bosque, dc=Dominio*... Se le llama

TABLA 1. Residencia de los NC's

COMPONENTE	DÓNDE RESIDE
NC de Dominio	En cada DC de ese Dominio.
NC de Esquema	En cada DC del Bosque.
NC de Configuración	En cada DC del Bosque.
Catálogo Global	Por defecto, un DC de cada site guarda el GC. Es configurable.

TABLA 2 . Diferencias entre modo nativo y mixto.

Acción	Modo Mixto	Modo Nativo
Replicación	El FSMO master DC envía cambios a los NTs BDCs, pero actúa como otro W2000 con los otros DCs.	Sólo hay Windows 2000, todos los DCs utilizan replicación multi-master
Autenticación	NT Lan Manager con los WNT y Kerberos en W2000.	Kerberos.
Localización RIDs	Centralizada.	Distribuida.
NetBIOS	No se puede desactivar.	Se puede desactivar.
Definiciones de grupos	Los globales no se pueden anidar y puede haber locales en los NT.	Se pueden empelar todos los grupos que permite Windows 2000.

el *Esquema*" y se replica en cada controlador de *Dominio* del *Bosque*. Guarda a lista de definiciones de objetos que se ueden crear en el *AD*. Para crear un uevo tipo de objeto, primero hay que lefinirlo en el *Esquema*.

ATÁLOGO GLOBAL

l *Catálogo Global* (*GC*) permite búsquedas rápidas, sin ser *NC*. Es una elección de las propiedades más útiles e cada objeto. Contiene una réplica arcial de cada *Dominio* del *Bosque* para úsquedas en el *Árbol*. Al hacer una consulta al *AD*, se rastrea primero el *GC*, y i no tiene la propiedad buscada, se consulta el *AD*, pero sólo en el *Dominio* actual (el *GC* es para el *Bosque*).

l GC permite búsquedas rápidas

Cada controlador de *Dominio* (*DC*) en n *Dominio* tiene una copia de la jerarquía del *Dominio* y del *Esquema* y *Configuración* del *Bosque*. Además, puede tener una copia del *GC* del *Bosque*. Dijimos ue el *Dominio* es el componente más equeño del *AD* reproducible en un *DC*, unidad más pequeña de partición. Significa que mientras todos los *Dominios* e *ARTIC.com* pueden estar en *DCs* separados, las *OUs* de cada *Dominio* no ueden partirse, y han de residir en todos los *DCs* del *Dominio* padre.

FSMO (FLEXIBLE SINGLE MASTER OPERATION)

l *AD* es una base de datos de directorio *Multimaster*. Esto implica que uede haber conflictos entre varios *aster*. En estos casos, el *Directorio Activo* selecciona un servidor para *Master*. n *W2000* hay cinco funciones (roles) ue tienen que ser efectivas en un solo ervidor. El servidor master de un rol n particular, es el propietario *FSMO* del rol (*FSMO role owner*). De los cinco roles, hay tres en el nivel de *Dominio* y dos en el nivel de *Bosque*. Cualquiera de los cinco roles se puede traspasar entre servidores por *MMC*. Si un servidor con un rol cae, hay que asignar el rol a otro servidor manualmente (maravillas de *Microsoft*). Los cinco *FSMO* roles son:

- *Master del Esquema*. Nivel de *Bosque*. Puede hacer cambios en el *Esquema*, y por defecto, primer servidor instalado en el *Bosque*.
- *Master de Nombres de Dominio*. Actúa en el nivel de *Bosque*. Puede tocar el contenedor de particiones y añadir o borrar *Dominios*. Es el primer *DC* instalado en el *Bosque* por defecto.
- *Anunciante PDC*. Nivel de *Dominio*. Por compatibilidad con los *Dominios NT*, un servidor *Windows 2000* debe funcionar como un *PDC* (controlador primario del *Dominio*). Este servidor es como un *NT master browser* y actúa como *PDC* para clientes y *BDCs*. Replica la base de datos *SAM* a los *BDCs*.
- *RID Master*. Nivel de *Dominio*. Cada objeto tiene un identificador de seguridad (*SID*) con el que el sistema identifica de forma única un objeto. Todos los objetos tienen el *GUID*, pero sólo tienen *SID* los que tienen seguridad. Se valida contra *SIDs* no *GUIDs*. El *RID Master* pone a disposición de los *DC* una lista de números para crear *SIDs*. Cada *DC* añadido pide al *RID Master* 100 valores libres para objetos que se creen en él. Cuando se le acaban pide más. Si cae este master y se le acaban los números a un *DC*, este *DC* no puede crear más objetos, y hasta que no se cambia el rol a un *DC* activo manualmente, no admite más *RIDs*.
- *Master de infraestructura*. Nivel de *Dominio*. Usado para mantener referencias a objetos en otros *Dominios*, los *Fantasmas* (o *Phantoms*). Por tanto, si tres usuarios del *Dominio B* son miembros de un grupo del *Dominio* A, el *Master de Infraestructura* del *Dominio* A se utiliza para

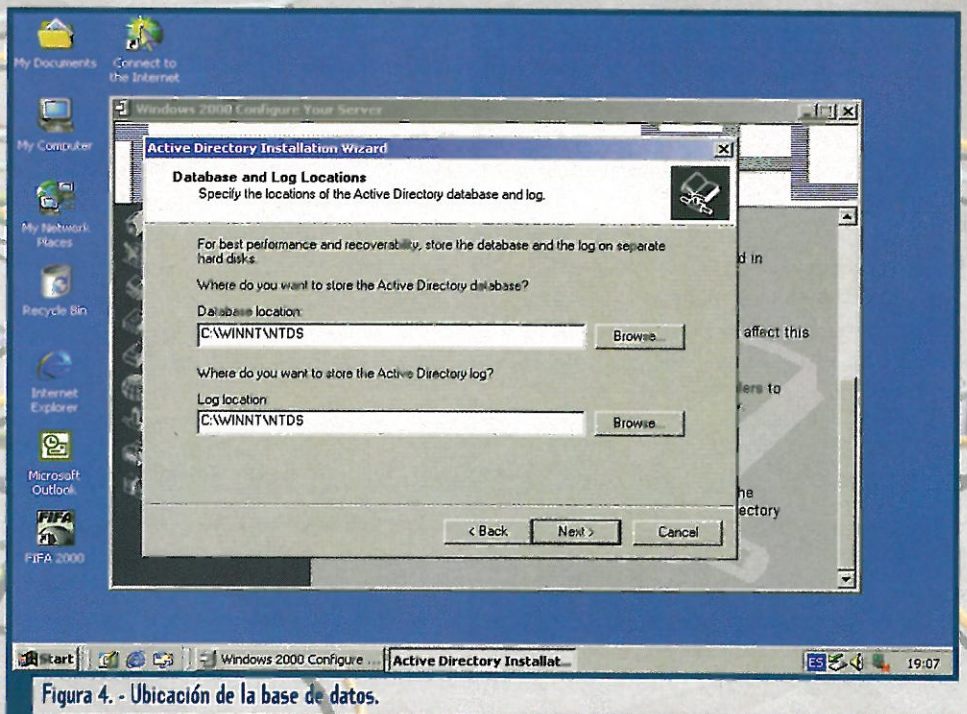


Figura 4. - Ubicación de la base de datos.

mantener las referencias a los usuarios del *Dominio Fantasma B*. El rol de *Master de Infraestructura* es vital para mantener continuamente los *links* a los *Fantasmas* cuando los objetos se mueven de un *Dominio* a otro. En un entorno con un solo *Dominio*, carece de función alguna .

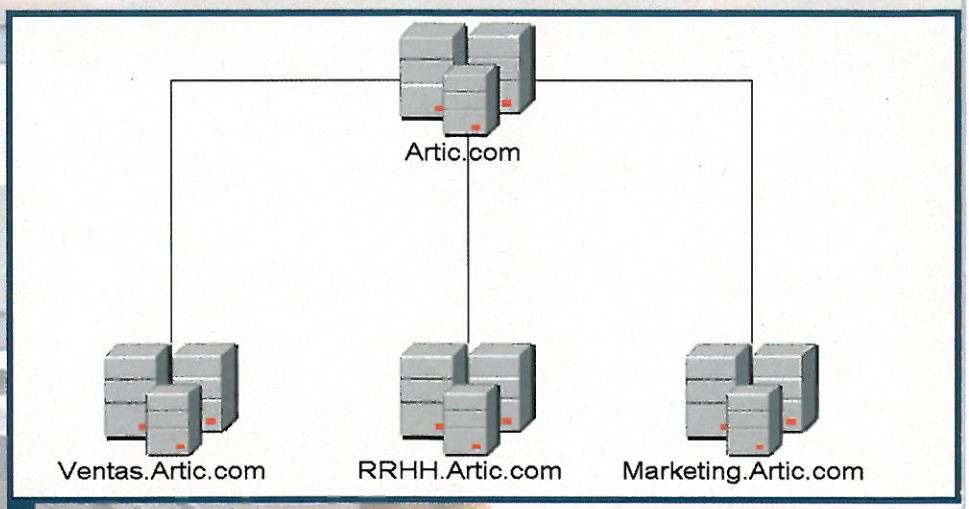


Figura 5. - Estructura de dominio.

MODO MIXTO Y MODO NATIVO

Cada *Dominio Windows 2000* puede estar en modo *mixto* (por defecto), o *nativo*. Un *Dominio* en *modo mixto* permite a los servidores con *NT* existir como parte del *Dominio*. Un *Dominio* nativo soporta sólo servidores *Windows 2000*. Recordemos que con las versiones previas de *Windows NT*, las redes tenían un *controlador primario* del *Dominio* (*PDC*) con una copia modificable de la base de datos de objetos (usuarios, grupos y servidores), y varios (o ningún) servidor de *Dominio* de *Backup* (*BDCs*) con copias de sólo lectura.

Los dominios pueden estar en modo mixto, o nativo

Entonces, en una red de servidores *Windows 2000*, para soportar a servidores *Windows NT*, un *Windows 2000* (y sólo uno) ha de actuar como *PDC*. Así, los servidores *BDCs* y clientes viejos que busquen un *PDC* encontrarán uno. Las diferencias entre modo mixto y modo nativo se muestran en la Tabla 4. El servidor que actúe de *PDC* sacará toda la información del *AD* necesaria para los clientes y *BDCs NT*, y la enviará por la red.

Pasar de *mixto* a *nativo* (*upgrade*) es sencillo. Simplemente conectas a un *DC* con el *Directory Manager Tool* y cambias las propiedades del *Dominio* de *mixto* a *nativo*. Aceptas, esperas a que todo se replique, y reinicias todos los *DCs* de ese dominio. Es un camino de una dirección, no hay vuelta atrás, a no ser que hagas un *backup*. No hagas el *upgrade* hasta que no estés seguro que no necesitas ningún servidor *BDC* en el *Dominio NT*.

Pasar un *Dominio mixto* a *nativo* sólo afecta a ese *Dominio*, y no a los otros, ya sean *raíz* o *subdominio*, pues sólo has quitado la capacidad de ese *Dominio* de replicar datos a servidores *Windows NT*, no la capacidad de réplica con servidores *W2000*.

GRUPOS

Windows 2000 soporta 3 tipos de grupos: *Local al Dominio*, *Global al Dominio*, y *Universal*. Cada uno de ellos se comporta diferente si está en *modo mixto* o en *modo nativo*. Cada clase de grupo puede tener dos *ámbitos* (*scopes*): *distribución* y *seguridad*. Si el *ámbito* es distribución, un grupo se puede definir como una lista de correo. En *Exchange* se llaman *Distribution Lists*, y el concepto es el mismo. Los grupos de seguridad pueden actuar como listas de correo, pero pueden tener *Access Control Lists* (*ACLs*) aplicadas de ellos a los objetos, ficheros o directorios del *Directorio Activo*.

Los grupos *locales* y *globales* proceden de los *grupos* de *Windows NT*, y se guardan en los *Dominios* donde se crean. Los grupos *universales* son los nuevos grupos de *Windows 2000*, y se guardan en el *Catálogo Global*. Los *globales* y *locales* actúan en el *Dominio*, y los *universales* en el *Bosque*.

En *Windows NT* los *Dominios* tenían dos tipos de grupos: *locales* y *globales*. Los *locales* podían contener usuarios y grupos *globales*. Los *globales* sólo podían contener usuarios. Ambos podían tener permisos. Los administradores solían poner los usuarios dentro de los *globales*, y luego meterlos en *locales* que tenían derechos sobre recursos para facilitar la administración.

En el AD caben 70 millones de objetos

Los grupos *locales* y *globales* de *NT* se mapean directamente sobre los *locales* y *globales* del *Directorio Activo* para un *Dominio*. Esto ocurre al exportar un *PDC NT* a *Windows 2000*, así los *BDCs* continúan viendo los grupos. Los grupos *universales* sólo están disponibles en modo nativo. Los otros grupos están disponibles en los dos *modos*. Lo complicado viene cuando queremos anidar grupos, ya que depende del tipo de grupo, modo y origen (*Dominio*).

Hay que recordar que los grupos **universales** *de seguridad* sólo existen en *modo nativo*, y los grupos **globales** *de seguridad* sólo pueden contener usuarios en *modo mixto*. Al pasar de *mo-*

TABLA 3 . Disponibilidad de grupos según modo.

Tipo	Ámbito	En mixto	En nativo
Dominio local	Seguridad	Si	Sí
Dominio global	Seguridad	Sí	Sí
Universal	Seguridad	No	Sí
Dominio local	Distribución	Sí	Sí
Dominio global	Distribución	Sí	Sí
Universal	Distribución	Sí	Sí

TABLA 4. Diferencias entre Windows NT y Windows 2000

WINDOWS NT	WINDOWS 2000
Réplica PDC a BDCs	Réplica multimaster entre DCs
Mínima unidad de partición: Dominio	Mínima unidad de partición: Dominio
Mínima unidad de autenticación: Dominio	Mínima unidad de autenticación: *OU*
Dominio como mínima unidad de política de sistema	*OU* como mínima unidad de política del sistema
Conexión y visor principal: *Broadcast NetBios*	Conexión y visor: *TCP/IP* al *AD*
WINS y *LMHOSTS* necesarios para *browsing* eficiente	*DNS* y *AD* necesarios para browsing eficiente
Objeto como mínima unidad de replicación	Propiedad como mínima unidad de replicación
Tamaño máximo de la *SAM*: 40 *Mb*	Tamaño máximo del *AD*: 70 *Tb*
Usuarios: 40.000	1/2 millones por Dominio, 10 por Bosque como máximo
4 modelos de Dominio: *(single, single-master, multimaster, complete-trust)*	El modelo es *complete-trust* (confianza total). Se puede modificar manualmente
El Esquema no se puede cambiar	El Esquema es modificable

do mixto a *nativo* no pierdes ninguna opción de anidamiento, y ganas:

- Los grupos *locales de seguridad* pueden contener *locales de seguridad* y grupos *locales de distribución*.
- Los grupos *globales de seguridad* pueden contener grupos *globales de seguridad* y *de distribución*.
- Los grupos de *seguridad universales* están disponibles.

Ya que los grupos *universales* están guardados en el *GC*, pueden ir más allá de los límites del *Dominio*, e incluirse en grupos *locales* o *universales* de otros *Dominios*. Una nueva habilidad del *modo nativo*, es que puedes convertir grupos de un tipo a otro. La conversión puede hacerse según lo que contenga en ese momento el grupo y el tipo y ámbito del grupo. Con el "*upgrade*" en *modo nativo* puedes convertir entre grupos:

- Los grupos de *seguridad* en grupos de *distribución*.
- Los grupos de *distribución* en grupos de *seguridad*.
- Un grupo *local* se puede convertir a *universal* siempre que el éste no sea miembro del otro grupo *local*.
- Un grupo *global* se puede convertir a *universal* si no contiene otro grupo *global*.

NT Y WINDOWS 2000

Los que hayan administrado *Windows NT* ya conocerán las grandes diferencias entre el *AD* de *W2000* y la *SAM* de *NT*, mostradas en la Tabla 6. Ya no hay *PDCs* y *BDCs*, sólo *DCs*. Puedes promover un servidor miembro normal a *DC*, y bajar un *DC* a miembro, sin reinstalar *Windows*. De hecho, cuando instalas un *Windows 2000 Server* o superior, se instala como *miembro*, no como *DC*. Con el *Wizard DCPROMO.EXE* lo promueves a *DC*. Te pregunta si quieres crear un nuevo *Árbol* o partir de uno existente, si quieres integrarlo en *Bosque* existente o crear uno nuevo, etcétera.

Como en *NT*, aún hay *Dominios*, pero ahora la unidad de autenticación y política más pequeña es la *OU*. Así se puede reducir el número de *Dominios*. *NT* usaba *NetBIOS* (sobre *TCP/IP*, *NetBEUI*, o *IPX*) como principal mecanismo de comunicación por red, mientras que *Windows 2000* está integrado con el *DNS* y usa *TCP/IP*. *W2000* ya no usa resolución de nombres *NetBIOS*, se basa en el *DNS* (adiós al *WINS*). El *WINS* se conserva para la compatibilidad con máquinas viejas. En *NT*, al cambiar un objeto, se replicaba por la red, y en *Windows 2000*, sólo el atributo.

El máximo recomendado de la *SAM* de *NT* eran 40 *Mb*, unos 40.000 objetos, aunque he visto *SAMs* de más de 75 *Mb* funcionando sin problemas. El *AD* de *Windows 2000* se guarda como al *Exchange*, con el "*Jet Blue database*". Caben hasta 10 millones de objetos por *Bosque*, hasta un tamaño de base de datos de 70 *Tb*. Recomiendan como máximo 2 millones de objetos por *Dominio*. Aunque ahora se guarda todo dentro del *AD*: usuarios, servidores, grupos, ordenadores, impresoras, contactos... El *Esquema* de *NT* no era ampliable. Ahora se pueden añadir objetos según necesidades.

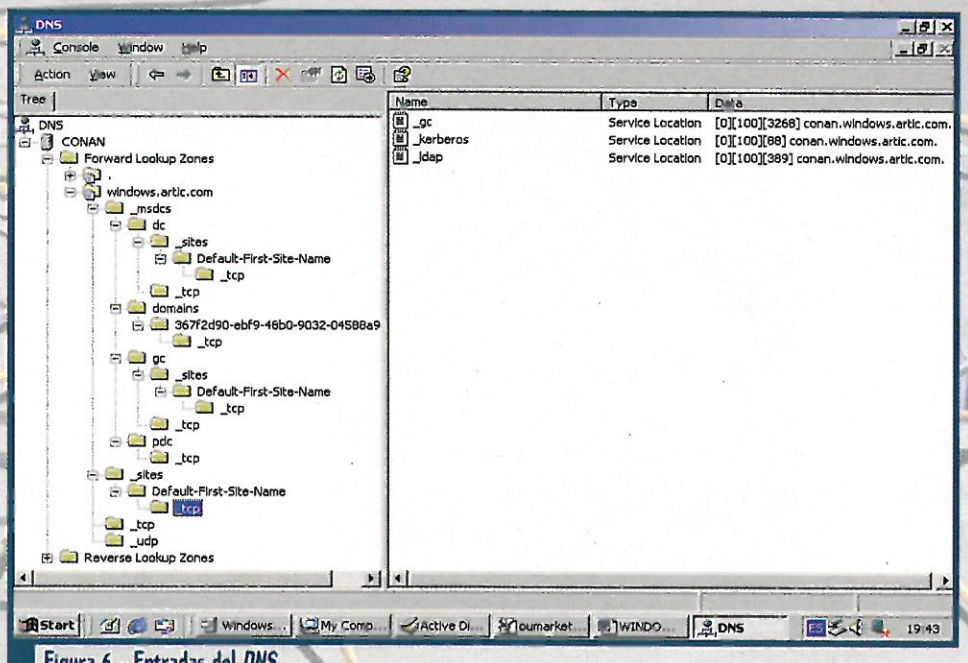


Figura 6. - Entradas del *DNS*.

RÉPLICA DEL DIRECTORIO ACTIVO

La réplica tiene dos aspectos, según cómo los datos son replicados entre *DCs* interconectados, y según cómo el *Knowledge Consistency Checker* (*KCC*) genera y mantiene los enlaces de replica entre servidores (ya dentro el mismo *site* o no). *Windows 2000* usa el termino *site* para definir una serie de máquinas que coexisten en una red rápida (red local) con buenas conexiones entre las secciones de esa red. *Windows 2000* usa los *sites* para replicar los datos por la red. La réplica de *Windows 2000* es muy eficiente, debido a que sólo replica propiedades, no objetos. Hay diferentes protocolos, todos sobre *TCP/IP* para replicar entre sites, cada uno se adapta a ciertas necesidades.

Los administradores deben crear los *sites*, no son automáticos, ya que *Windows 2000* no conoce tu red local. Una vez creado el *site*, un proceso de *Windows 2000* crea y programa enlaces de réplica *intrasite* entre los *DCs*. Al crear un segundo *site*, se iniciará la réplica entre sus *DCs*, pero no la *intersite*, que deberá configurarse manualmente.

¿Por qué tiene *sites Windows 2000*? Para que los clientes encuentren las cosas más rápido. Cuando conectas un portátil en un determinado *site*, tiene que ser capaz de encontrar un *DC*. Preguntará al *DNS* y al *AD*. Puede que un *site* no tenga servidores y tendrá que encontrar un *site* cercano. El *DNS* tiene los detalles de los sites, y el *AD* la topología. Así, sacará el *site* más cercano.

El AD es el modelo de todos los sistemas operativos

Windows 2000 puede replicar a través de enlaces muy lentos, incluso enviando mensajes por *e-mail* de forma asíncrona. Un *site* representa una zona de buena conectividad, pero no dicta nada de cómo estén conectados los sites.

Microsoft ha introducido bastantes términos nuevos. Los más elementales, necesarios para entender cómo se solucionan los conflictos entre modificaciones son:

- *Timestamp*. Fecha y hora guardada en el objeto para comparación.
- *USN*, valor de 64 *bits* asignado a un objeto que se incrementa cada vez que se modifica.
- *Property version number*. Número asociado a una propiedad que se incrementa cada vez que se modifica su valor.
- *Tombstone*. Los objetos no se borran inmediatamente, simplemente se marcan como borrados, y se eliminan al cabo de 60 días.
- *Globally Unique Identifier* (*GUID*). Cadena alfanumérica que representa un identificador único para un objeto.

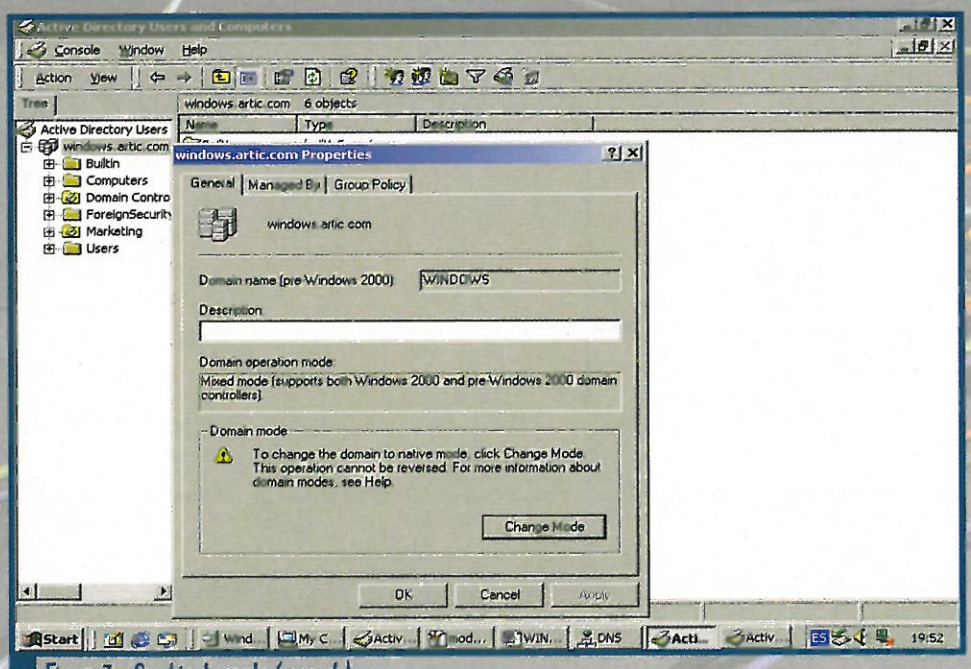


Figura 7. - Cambio de modo (upgrade).

Aunque el servicio de réplica funcione bien, a veces, los servidores hacen operaciones incompatibles entre ciclos de réplica. Esto ocasiona 3 tipos de conflictos:

- Conflicto por el cambio de la misma propiedad: se mira primero la versión de la propiedad en los dos objetos. La más alta, gana. En caso de empate, lo mismo con el *Timestamp* de los objetos. Si siguen empatados, miramos el *GUID*.
- Conflicto al mover un objeto a un recién borrado objeto padre: como el padre está borrado, se envía el objeto a un contenedor especial llamado *Lost and Found*. Su *AdsPath* es: *LDAP://cn=LostAndFound,dc=ARTIC,dc=com*.
- Conflicto debido a la creación de objetos con nombre iguales: el servidor busca el número de versión de los dos objetos. El que tiene versión más alta gana. Si no se mira el *Timestamp*, y si siguen empatados miramos el *GUID*. Los servidores van replicándose después entre ellos para arreglar los conflictos, se envían las modificaciones y las soluciones, hasta que todos esté en orden.

TCP/IP Y DNS

Windows 2000 se mueve sobre *TCP/IP*. El *AD* se comunica sólo por *TCP/IP*, y requiere soporte *DNS*. El *DNS* usado por el *AD* no es el clásico, si no *DDNS* (*DNS dinámico*). ¿Para qué quiere el *Directorio Activo* al *DDNS*? Para encontrar todos sus elementos. Cada vez que añades un *Dominio* al *AD* se hace un *update* en el *DDNS*. Cada vez que arranca una máquina cliente, se añade al *DDNS*, así que los clientes, servidores y *Dominios* necesitan el *DNS* para funcionar correctamente. Tienes dos opciones, o instalar un *DDNS* en la red para que las estaciones de trabajo, servidores y dominios modifiquen las entradas que requieran el *DDNS*, o modificar un *DNS*.

Para los que quieran utilizar *DNS*, hay información en la *web* de *Microsoft* que explica todas las entradas necesarias. Esto, en una red mediana puede ser un lío tremendo. Lo mejor es instalar un *DDNS open-source*, que aunque no abundantes, existen. De todas formas, escoger el que viene con *W2000*, puede ser lo más fácil.

Microsoft ha implementado unas normas de seguridad, *Secure Dynamic Update*, que requieren autenticación para modificar las entradas, y sólo los ordenadores

TABLA 5. Modificaciones.

	Se usa el DHCP de Microsoft sin Opción-81	Se usa el DHCP de Microsoft con Opción-81	Datos TCP/IP estáticos o servidor DHCP sin soporte DDNS
Cliente compatible DDNS (W98, W2000)	El cliente registra las entradas forward A y el servidor DHCP registra las entradas reverse PTR	El cliente registra las entradas forward (A y FQDN) y el servidor DHCP registra las entradas reverse PTR	El cliente registra toda la información al DDNS
Clientes de bajo nivel	El servidor DHCP registra todas las entradas forward A y reverse PTR	El servidor DHCP registra todas las entradas forward (A y FQDN) y reverse PTR	No hay registro automático

ue han creado una entrada pueden cambiarla. Además, las *ACLs* del *AD*.*Windows 000* permite integrar el *DNS* dentro del *AD*. Entonces, todos los *DCs* pueden resolver nombres. Y cada *DC* tiene una copia modificable del *DNS*, todos los *DCs* pueden actuar a la vez como servidores *DNS master* o *primario*. En una red con muchos *sites*, tener el *DNS* dentro el *AD* puede generar mucha más información de réplica. Por mucho que integres el *DNS*, siempre tendrás que nominar un servidor *DNS* como primario.

Curiosamente, *Microsoft* entiende que hay empresas donde se montará *Windows 2000* que tienen ya un *DNS*. Si todo funciona, no hay por qué tocarlo. Puedes asignar un *subdominio* al *DNS* de *Windows 2000*, por ejemplo *windows.artic.com*, e iniciar el *AD* y el *DDNS* desde este punto. Así, dejamos el artic.com a los servidores anteriores, y que el *DDNS* del *Windows 2000* controle a partir de *windows.artic.com*. Conozco una Universidad que funciona así, y de momento no han tenido problemas.

CLIENTES DE RED

Hablemos de los clientes. Si toda tu red con recientes servidores *Windows 2000* está formada por estaciones de trabajo *Windows 2000*, y *Windows 98*, todo funcionará perfectamente. Si tienes una mezcla de clientes de bajo nivel (*Downlevel Clients*) como *Windows 95*, *3.X*, *NT 3.X* y *NT 4.X*, tienes que saber que no funcionan correctamente con el *DDNS*. El *TCP/IP* de estos sistemas operativos no puede modificar el *DDNS* de *Microsoft*, sólo saben hacerlo los *98* y *2000*. Si hay un servidor *DHCP* compatible con *DDNS*, el servidor *DHCP* puede registrar los clientes de bajo nivel.

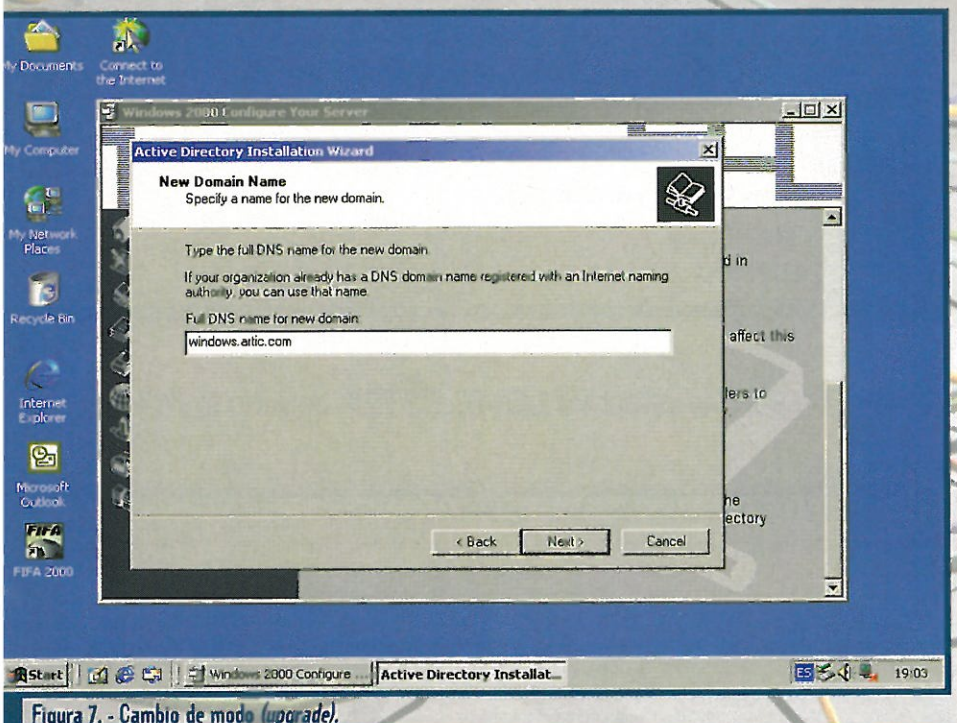


Figura 7. - Cambio de modo (upgrade).

La creación de sites no es automática

La especificación *DHCP* permite ser ampliada por un fabricante, y *Microsoft* ha sacado provecho de ello, al incluir la *Option-81* en el nuevo servidor de *DHCP, que p*ermite al cliente coger un *FQDN*, un nombre *DNS* absoluto, no relativo, como *estacion03.windows. artic .com* (ver Tabla 7).

CONCLUSIÓN

El *Directorio Activo* puede parecer complejo con tantos *Dominios*, *Bosques* y *Árboles*, pero es el camino que están siguiendo todos los sistemas operativos. *Windows 2000* incorpora algunas de las últimas mejoras del *TCP/IP* y del *DNS*. Introducir el *DNS* dentro el *Directorio Activo* ofrece una gran tolerancia a fallos y facilidad de administración, ya que no tienes que administrar el *DNS* en cada servidor. El *DDNS* puede ahorrar mucho trabajo a los administradores de redes. Si la estabilidad prometida por *Microsoft* resulta cierta, *Windows 2000* será un gran adversario frente otros sistemas operativos. Además *Microsoft* ha cuidado mucho la facilidad de programación. Las *Active Directory Services Interfaces* (*ADSI*), ampliamente documentadas, permitirán a terceros integrar sus aplicaciones con el *AD*.

Windows Scripting Host, ¿Herramienta o amenaza?

Marc Vaquer Crusat. *Técnico de Sistemas.*

El poder de *Windows Scripting Host* es grande. De su buen o mal uso depende que sea una excelente herramienta o una auténtica amenaza.

En el mes de mayo, el inquietante mensaje de correo electrónico *'I LOVE YOU'* se propagó por medio mundo, accediendo a los recursos del sistema e infectándolo. Además, se reenviaba a los contactos de la libreta de direcciones. Y sólo con un clic. ¿Cómo puede un simple archivo de texto atacar nuestro sistema? La respuesta es *Windows Scripting Host*.

Microsoft Windows Scripting Host (*WHS*)es una plataforma independiente de secuencias de comandos para sistemas *Windows* de 32 *bits*, distribuido con los motores de ejecución de los lenguajes *Visual Basic Script* y *JScript*, y admitiendo otros motores de lenguajes de *scripting*, como *Perl* o *Pitón* (también existen simuladores de páginas *ASP* para su ejecución con servidores *Apache* en *Linux*). Esta es una sencilla secuencia en *VBScript* mostrando en pantalla un mensaje de bienvenida: *WScript.echo Bienvenido a 'Windows Scripting Host'"*.

PROGRAMACIÓN Y ADMINISTRACIÓN

a tecnología ha pasado desapercibida cho tiempo, debido al uso exclusivo de estos lenguajes de *scripting* para la programación de páginas de contenido dinámico *Active Server Pages* (o *ASP*) en los servidores *Web* de *Microsoft*): *Internet Information Server* para *Windows NT* y *Personal Web Server* para *Windows 95/98*. En estos casos, la programación es hecha por un programador de aplicaciones *Web*, y la ejecución de los *scripts* es realizada únicamente por el servidor, cada vez que

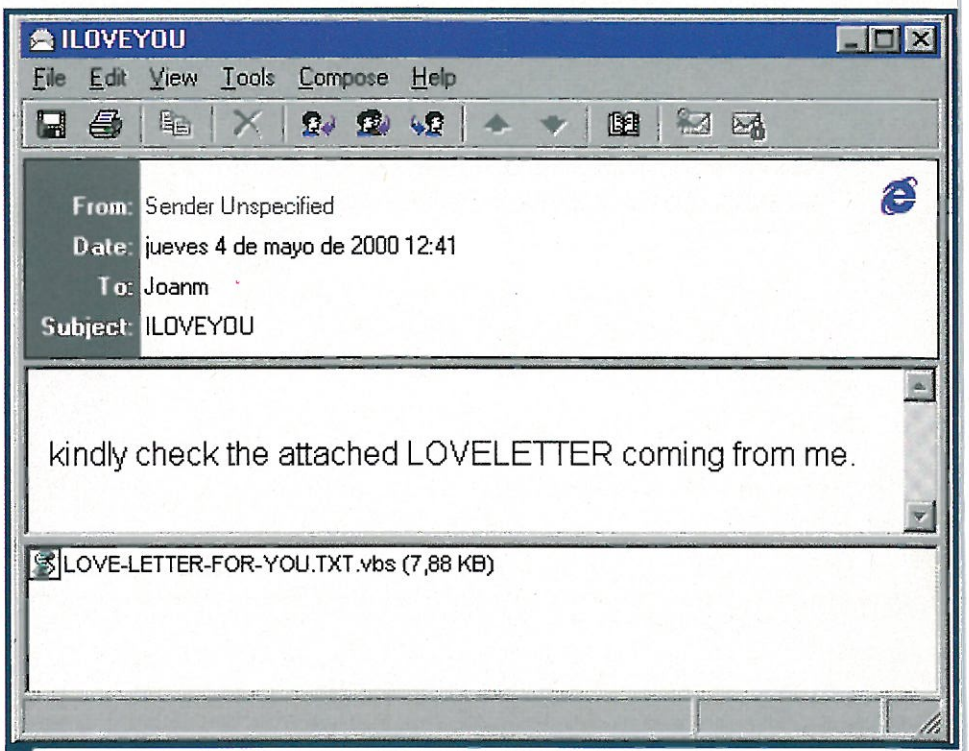


Figura 1. - Archivo VBS adjunto en un correo electrónico *'I LOVE YOU'*.

CUADRO 1. Características más importantes del objeto Wscript.

Propiedades	Descripción
Arguments	Colección de parámetros que accede WshArguments
ScriptName	String que contiene el nombre del archivo de secuencia de comandos que se está ejecutando.
ScriptFullName	String que contiene la ruta completa de la secuencia de comandos que se está ejecutando.
Métodos	**Descripción**
CreateObject	Crea un objeto especificado por el ProgID.
GetObject	Obtiene una referencia a un objeto a partir de un fichero existente.
Echo	Muestra el parámetro en la consola o ventana, según sea el caso.
Quit	Finaliza la ejecución con un código de error determinado.

CUADRO 2. Resumen de las características más importantes del objeto *WshShell*.

Propiedades	Descripción
Environment	Acceso al objeto *WshEnvironment*, que proporciona la colección de variables de entorno. Para *Windows 95* solamente podrá accederse a *'PROCESS'*, mientras que en *Windows NT 4.0* están disponibles *'SYSTEM'*, *'USER'*, *'VOLATILE'*, y *'PROCESS'*.
Métodos	**Descripción**
Run	Crea un nuevo proceso que ejecuta el comando que se recibe como parámetro.
PopUp	Muestra un cuadro de diálogo con un determinado texto durante un cierto número de segundos.

se solicita una página de un sitio *Web*. Así, tanto el desarrollo como la ejecución son controlados y seguros.

Los archivos de *scripts* también pueden ser ejecutados localmente por el usuario sobre su propio sistema. Al acceder directamente al registro del sistema y otros recursos importantes de éste, pueden desarrollarse *scripts* de configuración de equipos con relativa facilidad. Los administradores de redes locales pueden configurar los equipos de éstas utilizando archivos de *scripts* para *mapear* unidades de red, agregar impresoras compartidas, o crear accesos directos en el escritorio para aplicaciones distribuidas, etcétera.

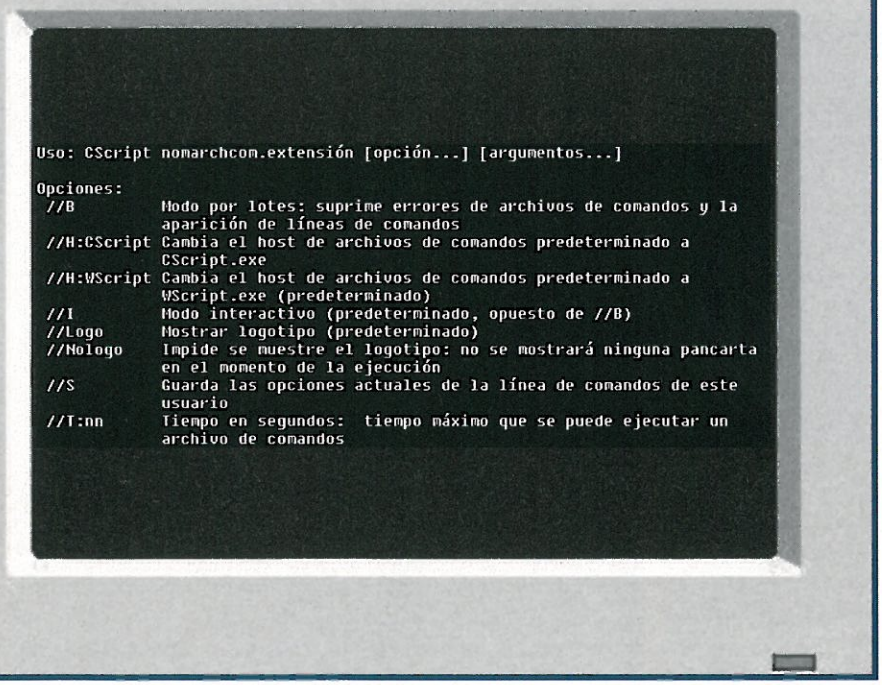


Figura 2. - Lista de propiedades de *CSCRIPT.EXE*.

Veamos un sencillo *script* que configura la *unidad Z*: sobre un recurso compartido de red:

```
Dim objNet
Set objNet = WScript.CreateObject("WScript.Net-
    work")
objNet.MapNetworkDrive "Z:", "\\Servidor\Carpeta-
    Compartida
```

CONSOLA O VENTANA

Los *scripts* pueden ejecutarse desde una ventana de comandos *DOS*, o desde las aplicaciones de *Windows*. El resultado se mostrará en la consola de comandos si se usa el intérprete *CSCRIPT.EXE*, o en ventanas con el intérprete *WSCRIPT.EXE*. En cualquier momento puede cambiarse el intérprete por defecto con las instrucciones:

```
wscript.exe //H:WScript
wscript.exe //H:CScript
cscript.exe //H:WScript
cscript.exe //H:Cscript
```

También podemos forzar la ejecución de una secuencia de comandos con un determinado intérprete agregando el nombre de su ejecutable en la línea de invocación del archivo: *'cscript.exe nomord.vbs'*, *script.exe nomord.vbs'*, o simplemente *'nomord.vbs'* para utilizar el intérprete por defecto.

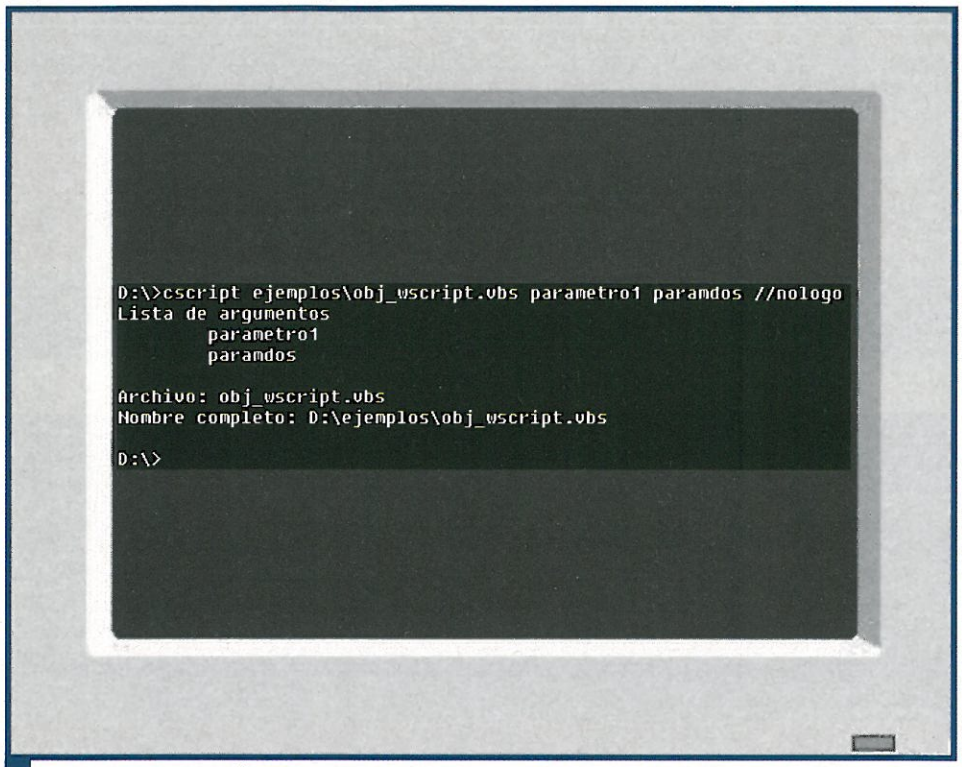


Figura 3. - Ejecución del listado 1 *CSCRIPT.EXE.*

PARÁMETROS DE EJECUCIÓN

Tanto *CScript* como *WScript* admiten una serie de modificadores. Introduciendo únicamente *CSCRIPT.EXE* en la línea de comandos de *DOS* obtenemos la lista de los parámetros que admite. Por otra parte, *WScript* tiene una hoja de propiedades a la que accedemos introduciendo *WSCRIPT.EXE* en la línea de comandos de *DOS*, o bien haciendo un clic con el botón derecho en los archivos de secuencias de comandos y seleccionando Propiedades. Las opciones *I* y *B* indican si el archivo de comandos debe producir una respuesta interactiva o no. Las opciones *LOGO* y *NOLOGO* indican si debe mostrarse o no la cabecera con el *copyright* de *Microsoft* y versión de *Windows Scripting Host*. El parámetro *T* o tiempo máximo de ejecución, según se trate de *CScript* o *WScript* respectivamente, indica los segundos que tiene el motor de secuencias de comandos para ejecutar y finalizar el *script*. Si transcurridos unos segundos el *script* no ha sido completamente ejecutado, se producirá un error en tiempo de ejecución.

Por último, la opción *S* guarda las opciones seleccionadas en el perfil del usuario.

Viendo todas las posibilidades de ejecución de estos archivos de comandos, comprobamos que no existe configuración alguna sobre su seguridad o uso, y que por lo tanto, si las acciones programadas no necesitan la confirmación del usuario, éstas serán llevadas a cabo sin concesión alguna, aunque perjudiquen al propio sistema.

MODELO DE OBJETOS

Windows Scripting Host se compone de diferentes archivos. *CScript.exe / WScript.exe* aporta el objeto *Wscript* y la clase *WshArguments*. WSHom.ocx aporta los objetos de automatización *WshShell* y *WshNetwork*, y las clases *WshShortcut, WshUrlShortcut, WshCollection, WshEnvironment* y *WshSpecialFolders*.

Las clases son accesibles como propiedades de los objetos, que deberán haber sido creados antes. La única excepción es el objeto *Wscript*, que está directamente expuesto por el motor de

LISTADO 1. Acceso a objetos de aplicaciones del sistema (CREAOBJ.VBS).

```
Dim objFS, objFile
Dim strText
Set objFS = WScript.CreateObject("Scripting.FileSystemObject")
set objFile = objFS.CreateTextFile("c:\logfile.txt",true)
strText = "Empezó a las " & Time()
wscript.echo strText
objFile.WriteLine(strText)
wscript.echo "Código adicional"
strText = "El texto se escribe directamente sobre el fichero"
objFile.WriteLine(strText)
strText = (11.6^10*4.32432)/14323.23^2
objFile.WriteLine(strText)
strText = "Finalizó a las " & Time()
wscript.echo strText
objFile.WriteLine(strText)
objFile.Close
set objFile = nothing
set objFS = nothing
```

LISTADO 2. Acceso a objetos con eventos (EVENTO.VBS).

```
Dim objIE
Dim xstr
Dim bolExit
Set objIE = WScript.CreateObject("InternetExplorer.Application","IE_")
Wscript.echo "Aplicación cargada."
bolExit = false
objIE.navigate "http://www.servidor.www/default.asp"
MsgBox "Pulsa para descargar la aplicación"
objIE.Quit
Set objIE = Nothing
Wscript.echo "Aplicación descargada."

sub IE_DocumentComplete(ByVal pDisp, URL)
    bolExit = true
    xstr = objIE.Document.Title
    Wscript.echo "El documento '" & xstr & "' ha sido cargado."
end sub
```

LISTADO 3 . Ejemplo de propiedades y métodos expuestos en los Cuadros 2 y 3.

```
Dim WshShell
Dim WshNetwork
Dim strEnv
Dim strText

Set WshShell = WScript.CreateObject("WScript.Shell")
strText = "El cuadro de diálogo desaparecerá en 10 segundos." &
vbCrLf
strText = strText & "Las variables de entorno aparecerán a continuación."
WshShell.PopUp strText,10
strText = ""
For Each strEnv In WshShell.Environment("PROCESS")
    strText = strText & strEnv & vbCrLf
Next
Wscript.echo strText

Set WshNetwork = WScript.CreateObject("WScript.Network")
strText = "Usuario " & WshNetwork.UserName & " en el equipo "
& WshNetwork.ComputerName
Wscript.echo strText

if MsgBox("¿Quieres ejecutar otra vez este script?",vbYesNo)=vbYes
then
    WshShell.Run Wscript.ScriptFullName
end if
```

LISTADO 4 . Ejemplo de acceso al registro.

```
Dim WSHShell
Set WSHShell = WScript.CreateObject("WScript.Shell")

WScript.Echo "Creando la clave HKCU\Ejemplo con el valor 'Primer valor'"
WSHShell.RegWrite "HKCU\Ejemplo\", "Primer valor"

WScript.Echo "Crear la clave HKCU\Ejemplo\Entrada con el valor 'Segundo valor'"
WSHShell.RegWrite "HKCU\Ejemplo\Entrada\", "Segundo valor"

WScript.Echo "Establecer el valor HKCU\Ejemplo\Valor a REG_SZ 1"
WSHShell.RegWrite "HKCU\Ejemplo\Valor", 1

WScript.Echo "Establecer el valor HKCU\Ejemplo\Entrada a REG_DWORD 2"
WSHShell.RegWrite "HKCU\Ejemplo\Entrada", 2, "REG_DWORD"

WScript.Echo "Establecer el valor HKCU\Ejemplo\Entrada\Valor a REG_BINARY 3"
WSHShell.RegWrite "HKCU\Ejemplo\Entrada\Valor", 3, "REG_BINARY"

WScript.Echo "Eliminar el valor HKCU\Ejemplo\Entrada\Valor"
WSHShell.RegDelete "HKCU\Ejemplo\Entrada\Valor"

WScript.Echo "Eliminar la clave HKCU\Ejemplo\Entrada"
WSHShell.RegDelete "HKCU\Ejemplo\Entrada\"

WScript.Echo "Eliminar la clave HKCU\Ejemplo"
WSHShell.RegDelete "HKCU\Ejemplo\"
```

ecuencias de comandos, y que consta e las propiedades y métodos detallados n el Cuadro 1.

A partir del objeto *Wscript* podemos cceder al entorno de ejecución del pro-

pio archivo de comandos, tal y como se indica:

```
Dim strText
Dim strArg
strText = "Lista de argumentos" & vbCrLf
for each strArg in Wscript.Arguments
    strText = strText & vbTab & strArg & vbCrLf
next
strText = strText & vbCrLf & "Archivo: " &
    Wscript.ScriptName
strText = strText & vbCrLf & "Nombre completo: " &
    Wscript.ScriptFullName
Wscript.Echo strText
Wscript.Quit
```

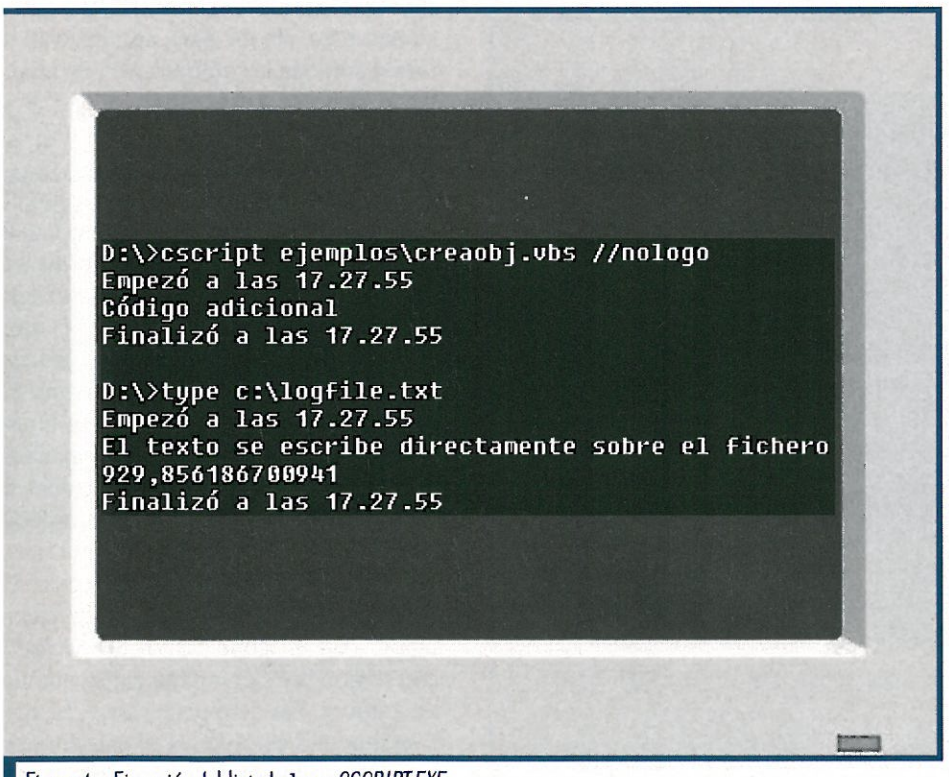


Figura 4. - Ejecución del listado 1 con *CSCRIPT.EXE*.

CREACIÓN Y DESTRUCCIÓN DE OBJETOS

Los métodos *CreateObject* y *GetObject* pueden concretar cualquier objeto que esté registrado en el sistema donde es ejecutado. Las clases concretables se encuentran en la ramificación *HKEY_CLASSES_ROOTS* del registro de *Windows*. Una vez se ha creado el objeto, puede accederse a las propiedades y métodos de éste.

El listado 1 detalla el acceso al sistema de archivos con el objeto *FileSystemObject*, que tiene exactamente la misma funcionalidad que en las páginas activas de servidor o *ASPs*, en *VBScript*.

Los métodos *CreateObject* y *GetObject* admiten como parámetro un *string* que será el prefijo de los procedimientos de evento que active el objeto. Por ejemplo, si el objeto tiene el evento *On-*

CUADRO 3. Resumen de las características más importantes del objeto *WshNetwork*.

Propiedades	Descripción
ComputerName	Nombre del equipo.
UserDomain	Nombre del dominio en el que se inició la red (si corresponde).
UserName	Nombre de usuario.

CUADRO 4. Métodos del objeto WshShell para el acceso al registro

Métodos	Descripción
RegDelete	Elimina una clave o valor del registro.
RegRead	Recupera una clave o valor.
RegWrite	Crea una nueva clave o valor en el registro.

Change y se agrega el prefijo *Obj_* en la sentencia de su creación, el procedimiento *Obj_OnChange* se ejecutará cuando se active el evento para la concreción del objeto.

Como puede verse en el listado 2, *WSH* permite definir procedimientos de eventos de los objetos concretados, con lo que el programador consigue todavía más control sobre éstos. En el ejemplo puede verse cómo el archivo de comandos usa una concreción de *Microsoft Internet Explorer* para descargar una página de un servidor y mostrar un mensaje con el título de ésta.

OBJETOS WSHSHELL Y WSHNETWORK

A partir del objeto *Wscript* y mediante el método *CreateObject* pueden invocarse los objetos de automatización *WshShell* y *WshNetwork*. Estos dos objetos proporcionan propiedades y métodos para acceder a cualquier componente del sistema: registro, red local y carpetas especiales.

El Listado 3 muestra el acceso a las variables de entorno del sistema, así como los nombres del equipo, usuario y dominio. Por otra parte, es posible ejecutar cualquier aplicación de la que se conozca su ruta de acceso, incluida la propia secuencia de comandos.

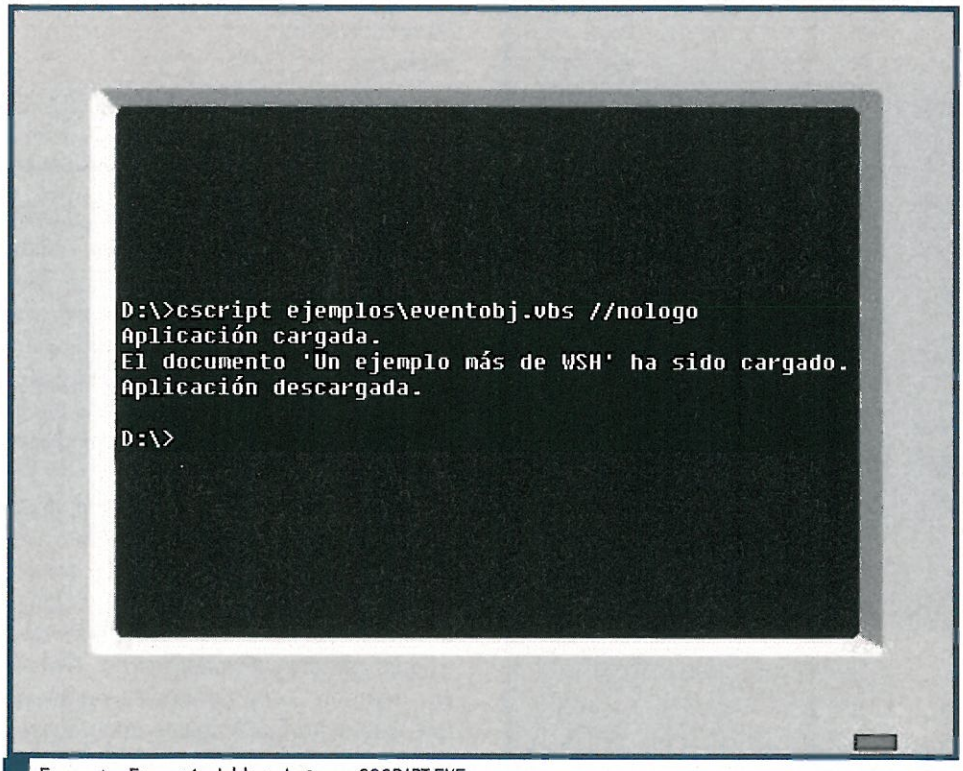


Figura 4. - Ejecución del listado 2 con *CSCRIPT.EXE*.

EL REGISTRO

Los métodos de acceso al registro son proporcionados por el objeto *WshShell* y se resumen en el Cuadro 4. Las rutas del registro deben delimitarse con el carácter "\", y deben comenzar por una raíz válida:
HKCU - HKEY_CURRENT_USER,
HKLM - HKEY_LOCAL_MACHINE,
HKCR - HKEY_CLASSES_ROOT.

El acceso directo al registro es quizá el componente más peligroso de *Windows Scripting Host*, aunque muy útil, pues almacena multitud de configuraciones, tanto *hardware* como *software*. Se usa en archivos de comandos de configuración: para *ODBCs*, acceso a *Internet*, acceso a *Impresoras*, para escribir directamente la configuración adecuada, etcétera. Basta conocer la ruta de registro donde ubicar la información.

LA RED

Los métodos para acceder a los componentes de la red son proporcionados por el objeto *WshNetwork*, y se resumen en el Cuadro 5. Mediante éstos, pueden manipularse las unidades *mapeadas* de red y las conexiones a impresoras. Igual que el registro, *WSH* proporcionar este objeto para administrar de forma cómoda la configuración de unidades e impresoras de red.

CARPETAS Y ARCHIVOS

El objeto *WshShell* proporciona la propiedad *SpecialFolders* como colección de rutas de acceso a carpetas especiales como el escritorio, los favoritos, etcétera, y un método *CreateShortcut* para la creación de archivos de acceso directo. Por otra parte, el objeto de sistema *FileSystemObject* proporciona acceso al sistema archivos, incluyendo creación, eliminación, lectura, modificación, cambio de nombre y atributos de ficheros y carpetas.

Por ejemplo, el Listado 5 muestra cómo crear en el escritorio un acceso directo al propio *script* que se está ejecutando, con el icono de un ordenador y con el nombre del equipo como nombre de archivo. Por otra parte, en el Listado 2 se muestra cómo crear el objeto de acceso al sistema de archivos y cómo operar con él. *FileSystemObject* proporciona

LISTADO 5. Ejemplo de creación de acceso directo en el escritorio del usuario.

```
Dim WshShell
Dim objShortcut
Dim strPath

Set WshShell = WScript.CreateObject("WScript.Shell")
strPath =WshShell.SpecialFolders("Desktop") & "\" & WSHShell.ExpandEnvironmentStrings("%computername%") & ".lnk"
Set objShortcut = WSHShell.CreateShortcut(strPath)

objShortcut.TargetPath = WSHShell.ExpandEnvironmentStrings(WScript.ScriptFullName)
objShortcut.WorkingDirectory = WSHShell.ExpandEnvironmentStrings("%windir%")
objShortcut.WindowStyle = 4
objShortcut.IconLocation = WSHShell.ExpandEnvironmentStrings("%windir%\explorer.exe, 0")
objShortcut.Save
```

CUADRO 5. Métodos del objeto WshNetwork para el acceso a componentes de red.

Métodos	Descripción
EnumNetworkDrives	Colección de unidades *mapeadas* en red.
MapNetworkDrive	Asigna una nueva letra de unidad a un recurso compartido en la red.
RemoveNetworkDrive	Elimina una unidad *mapeada* de red.
EnumPrinterConnections	Colección de impresoras compartidas configuradas en el sistema.
AddPrinterConnection	Agrega una nueva impresora compartida.
RemovePrinterConnection	Elimina una impresora compartida del sistema.
SetDefaultPrinter	Establece el parámetro como impresora por defecto.

la propiedad *Drives* y los métodos:

para acceder a archivos: *FileExists, DeleteFile, CopyFile, MoveFile, CreateTextFile, OpenTextFile, GetFile, GetFileName* y *GetExtensionName.*

para acceder a carpetas: *FolderExists, DeleteFolder, CopyFolder, MoveFolder, CreateFolder, GetFolder, GetParentFolderName, GetSpecialFolder* y *GetFileName*.

Para acceder a unidades de disco: DriveExists, GetDrive y *GetDriveName*.

Para trabajar con rutas de acceso: *BuildPath, GetAbsolutePath, GetBaseName* y *GetTempName.*

El acceso al sistema de archivos es tan peligroso como el acceso al registro. Debe usarse de una forma lógica, y no acceder a archivos ajenos a los propósitos de la aplicación o de la configuración, puesto que no habrá solicitud de confirmación de los cambios.

CONCLUSIÓN

Windows Scripting Host es una herramienta poderosa. Pero puede convertirse en un arma con un poder de destrucción enorme a la espera de un clic realizado por alguien sin conocimiento de las consecuencias de la ejecución de la secuencia de comandos. Como ha podido verse en estas páginas, sólo depende del programador realizar un buen uso de esta tecnología. ¿Una herramienta, o una amenaza? Tú decides.

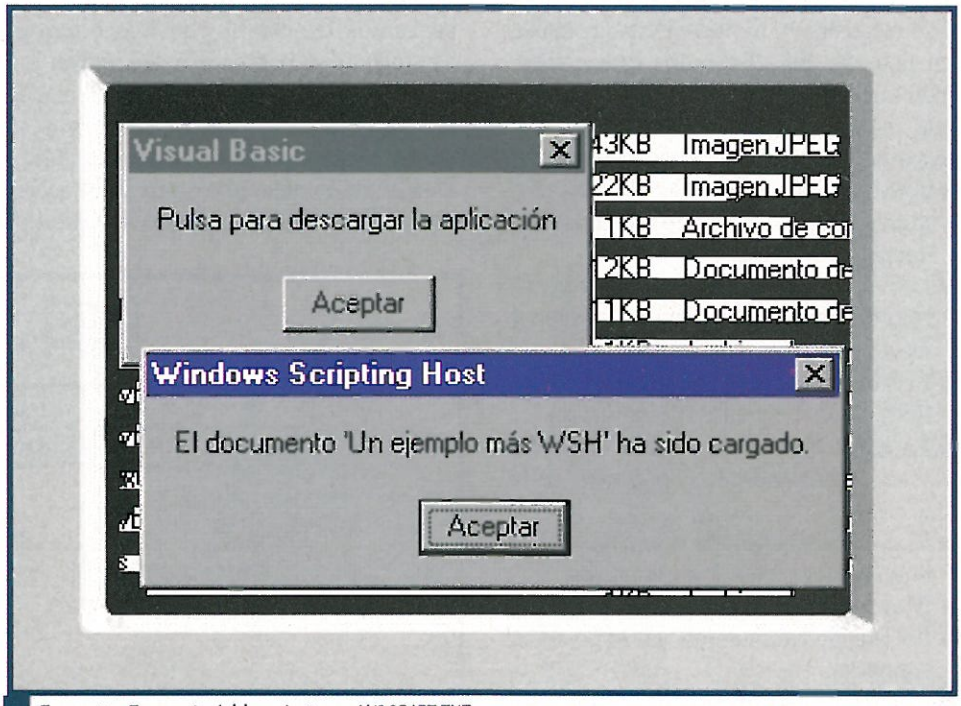
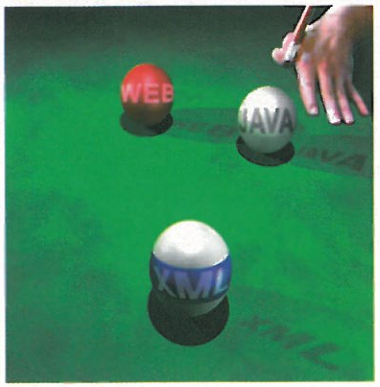


Figura 5. - Ejecución del listado 2 con *WSCRIPT.EXE.*



XNL / Java (II)

Cada cosa en su sitio

Adolfo Aladro García. *Analista Programador*

Document Object Model describe la estructura interna de un documento *XML* y la forma de obtener su información. El *API JAXP* ofrece eficientes interfaces de acceso al *DOM*.

ESTRUCTURACIÓN

En el capítulo interior se hizo una introducción al lenguaje *XML*, así como al desarrollo de los documentos *DTD* que garantizan la validez de los *XML*. La estructuración de fuentes de datos es uno de los aspectos más importantes del desarrollo de aplicaciones de tecnología *XML*. Supongamos que queremos representar con *XML* las noticias de un periódico electrónico. El primer paso consiste en detectar los elementos que conforman una noticia y estudiar la forma en la que éstos se agrupan. Así, podríamos desglosar una noticia con el criterio:

- Toda noticia tiene un identificador único que las distingue de todas las demás.
- Los elementos de una noticia son: título, fecha, autor, resumen, cuerpo y fotografías.
- El título es la cadena de texto que identifica, describe y sintetiza la noticia.
- La fecha tiene tres valores: día, mes y año, referidas a la confección de la noticia.
- El autor corresponde a la fuente o el nombre del redactor de la noticia.
- El resumen es la exposición sintética.
- El cuerpo representa la exposición completa.
- Finalmente, las fotografías o ilustraciones que completan la noticia.
- Cada fotografía se define con el nombre del fichero donde está almacenada y el pie, o la frase que aparece debajo de la fotografía.

El tipo de un nodo determina su valor y su nombre

La Figura 1 muestra la estructura de una noticia, según los criterios que acabamos de establecer. Los elementos marcados con azul representan los atributos. Todas las entidades se agrupan formando una estructura arborescente típica de los documentos *XML*. Desde este análisis, es muy fácil escribir el código *XML* que represente una noticia. El Listado 1 ofrece un ejemplo de noticia.

No debemos olvidar las normas que hacen que un documento *XML* esté bien formado. Todas las etiquetas deben tener su correspondiente etiqueta de cierre: *<resumen> ··· </resumen>*, o terminar con la barra inclinada (/) para significar que son etiquetas sin etiqueta de cierre: *<fecha dia="05" mes="06" ano="2000"/>*. El valor de los atributos debe estar entrecomillado: *<noticia id="3578"> ··· </noticia>*. Además hay que utilizar las entidades de carácter para representar los caracteres especiales, como por ejemplo las vocales acentuadas: *<titulo>UN TEMA INÉDITO DE MADONNA, DISPO-*

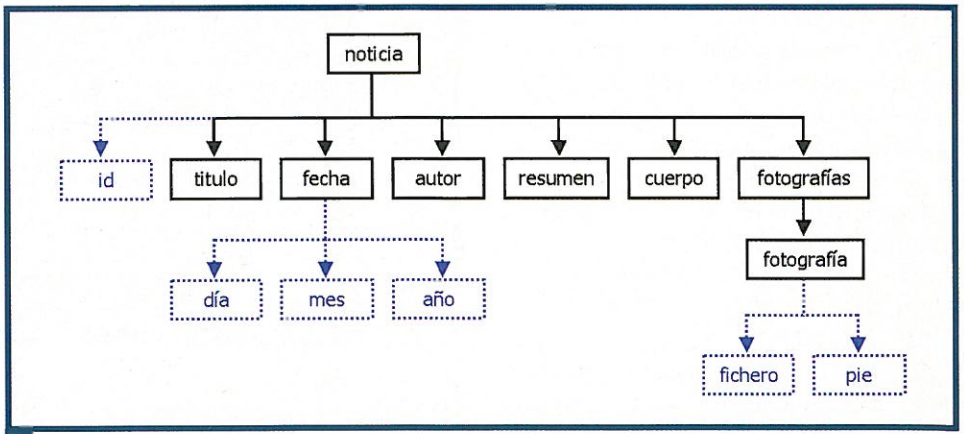


Figura 1. - Estructura de una noticia.

IBLE EN LA RED</titulo>. El fiche- *noticia.xml* contiene el código com- eto de la noticia estructurada en forma origen de datos *XML*.

DOCUMENTO DTD

l siguiente paso consiste en crear el documento *DTD* asociado a la fuente datos *XML*. Esto resulta muy fácil, sta con seguir paso a paso lo expues- en los puntos arriba mencionados.

Los elementos que conforman una noticia:

```
ELEMENT noticia (titulo,fecha,au-
   tor,resumen,cuerpo,fotografias)>
```

Toda noticia cuenta con un identificador único:

```
ATTLIST noticia id CDATA #REQUIRED>
```

El título es la cadena de texto que identifica:

```
ELEMENT titulo (#PCDATA)>
```

La fecha tiene tres valores:

```
ELEMENT fecha EMPTY>
ATTLIST fecha dia CDATA #REQUIRED>
ATTLIST fecha mes CDATA #REQUIRED>
ATTLIST fecha ano CDATA #REQUIRED>
```

El autor es el nombre del responsable de redactar la noticia:

```
ELEMENT autor (#PCDATA)>
```

El resumen es la exposición sintética de la noticia:

```
ELEMENT resumen (#PCDATA)>
```

El cuerpo representa la exposición completa de la noticia:

```
ELEMENT cuerpo (#PCDATA|BR)*>
ELEMENT BR EMPTY>
```

Las noticias contenidas en el archivo *ML* están destinadas a formar parte una página *HTML*, por lo que en el erpo encontramos etiquetas *
*. stás también deben conformar el do- mento *DTD*, como acabamos de ha- r para garantizar la validez del origen datos.

Las fotografías que completan la noticia:

```
ELEMENT fotografias (fotografia)+>
```

Cada fotografía se define por el nombre del fichero:

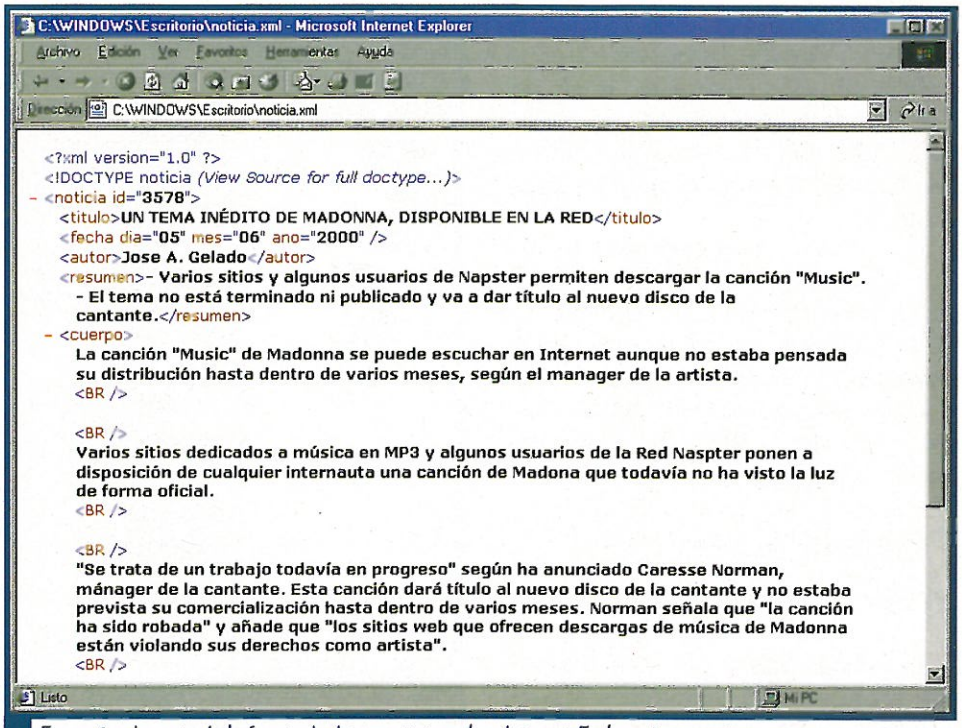


Figura 2. - Aspecto de la fuente de datos *noticia.xml* en *Internet Explorer 5.x.*

```
<!ELEMENT fotografia EMPTY>
<!ATTLIST fotografia fichero CDATA
    #REQUIRED>
<!ATTLIST fotografia pie CDATA #REQUI-
    RED>
```

La estructuración es uno de los pasos más importantes

El archivo *noticia.dtd* guarda el código fuente del documento *DTD* recién creado. Para vincular este documento al origen de datos *XML* basta con añadir al archivo noticia*.xml*:

```
<!DOCTYPE noticia SYSTEM
    "noticia.dtd">
```

INTRODUCCIÓN AL DOM

El paquete *ord.w3c.com*, que forma parte del *API JAXP*, contiene las interfaces para acceder al *DOM* de un origen de datos *XML*. Hay muchas formas de recorrer los datos contenidos en un documento *XML*, por lo que la programación puede resultar un poco confusa al principio. Lo importante es entender claramente la estructura arborescente construida a partir de la fuente de datos *XML*, y cuáles son las características que diferencian a unos nodos de otros.

Las interfaces del paquete *ord.w3c .com* son: *Attr, CDATASection, CharacterData, Comment, Document, Document Fragment, DocumentType, DOMImplementation, Element, Entity, EntityRefe*

LISTADO 1. Código resumido de noticia.xml.

```
<?xml version="1.0"?>
<noticia id="3578">
  <titulo>UN TEMA IN&#201;DITO DE MADONNA, DISPONIBLE EN LA RED</titulo>
  <fecha dia="05" mes="06" ano="2000"/>
  <autor> ... </autor>
  <resumen> ... </resumen>
  <cuerpo> ... </cuerpo>
  <fotografias>
    <fotografia fichero="4d3d7020818i3.gif" pie="Madonna"/>
  </fotografias>
</noticia>
```

rence, NamedNodeMap, Node, NodeList, Notation, ProcessingInstruction, y *Text.* Cada una de ellas representa un tipo de elemento de la estructura de un documento *XML*. El Listado 2 contiene un esquema del árbol que agrupa a todas estas interfaces.

La interfaz *Node* representa una abstracción global de un nodo del árbol conformado a partir de un documento *XML*. La mayor parte del resto de las interfaces que heredan de ésta, no son más que particularizaciones que se hacen según las características específicas de cada nodo.

El documento DTD debe partir del modelo desarrollado

La interfaz *Node* tiene muchos métodos, clasificados en tres grupos dependiendo de su funcionalidad: los que proporcionan información del nodo; aquellos que sirven para recorrer el árbol de nodos; y finalmente, los métodos que permiten crear dinámicamente fuentes de datos *XML*. En el primero destacan *public short getNodeType(), public java.lang.String getNodeName(), y public java.lang.String getNodeValue() throws DOMException.*

El método *getNodeType* devuelve un número entero que representa el tipo de nodo. La interfaz *Node* tiene las definiciones *public static final short ELEMENT_NODE* y *public static final short ATTRIBUTE_NODE.*

Es muy fácil elaborar una función que, a partir del valor numérico devuelto por el método *getNodeType,* proporcione una cadena de texto representando el tipo del nodo, lo que sin duda, es de más utilidad.

```
public static String getNodeTypeByNa-
    me(int n) {
    if (n == Node.ELEMENT_NODE) {
        return "ELEMENT_NODE";
    } else if (n ==
    Node.ATTRIBUTE_NODE) {
        return "ATTRIBUTE_NODE";
    } else if (n == Node.TEXT_NODE) {
        return "TEXT_NODE";
    } else if (n == Node.CDATA_SEC-
    TION_NODE) {
        return "CDATA_SECTION_NODE";
    } else if (n == Node.ENTITY_RE-
    FERENCE_NODE) {
        return "ENTITY_RE-
    FERENCE_NODE";
    } else if (n == Node.PRO-
    CESSING_INSTRUCTION_NODE)
    {
        return
    "PROCESSING_INSTRUCTION_NO-
    DE";
    } else if (n == Node.COM-
    MENT_NODE) {
        return "COMMENT_NODE";
    } else if (n == Node.DOCU-
    MENT_NODE) {
        return "DOCUMENT_NO-
    DE";
    } else if (n == Node.DOCU-
    MENT_TYPE_NODE) {
        return "DOCUMENT_TY-
    PE_NODE";
    } else if (n == Node.DOCU-
    MENT_FRAGMENT_NODE) {
        return "DOCUMENT_FRAG-
    MENT_NODE";
    } else if (n == Node.NOTA-
    TION_NODE) {
        return "NOTATION_NO-
    DE";
    } else {
        return "UNKNOWN";
    }
}
```

El método *getNodeName* facilita el nombre del nodo correspon-

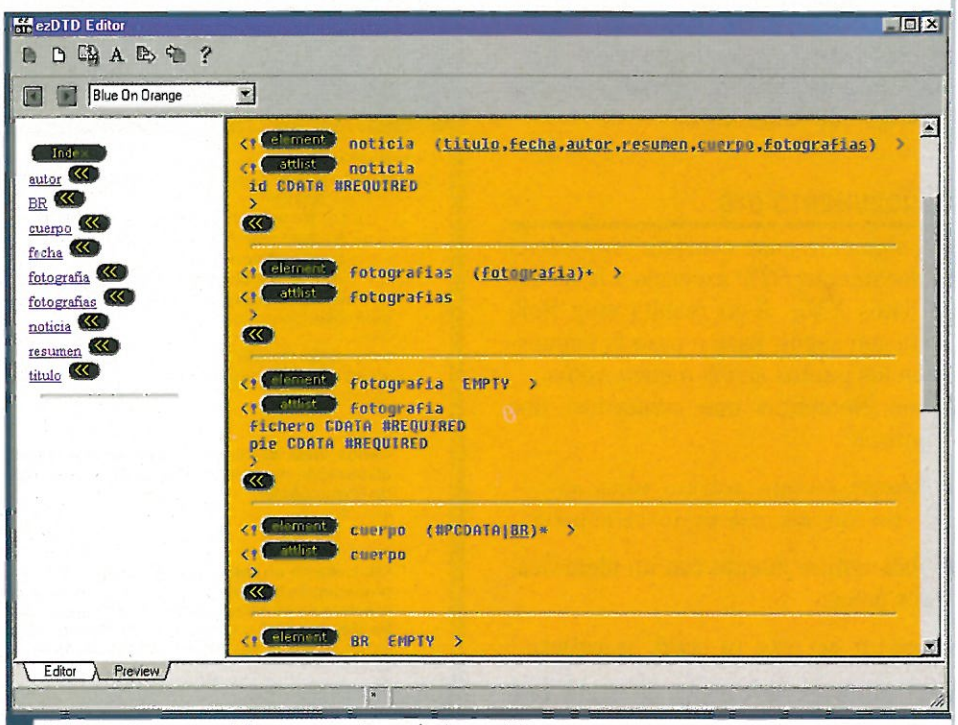


Figura 3. - Documento *DTD* representado por el editor *ezDTD.*

LISTADO 2 . Interfaces del paquete org.w3c.dom.

org.w3c.dom.DOMImplementation
org.w3c.dom.NamedNodeMap
org.w3c.dom.Node
- *org.w3c.dom.Attr*
- *org.w3c.dom.CharacterData*
- *org.w3c.dom.Comment*
- *org.w3c.dom.Text*
  - *org.w3c.dom.CDATASection*
- *org.w3c.dom.Document*
- *org.w3c.dom.DocumentFragment*
- *org.w3c.dom.DocumentType*
- *org.w3c.dom.Element*
- *org.w3c.dom.Entity*
- *org.w3c.dom.EntityReference*
- *org.w3c.dom.Notation*
- *org.w3c.dom.ProcessingInstruction*

org.w3c.dom.NodeList

liente. Los nombres de un nodo del árbol *XML* dependen de de su tipo. Así, cuando un nodo es del tipo *ELEMENT_NODE*, el valor devuelto por el método *getNodeName* es el nombre de la etiqueta *XML*; cuando un nodo es del tipo *TEXT_NODE*, el valor devuelto es la cadena *#text*. A medida que nos familiaricemos con el *DOM* sabremos con exactitud cuál es el nombre esperado para cada tipo distinto del nodo.

El método *getNodeValue* proporciona el valor del nodo. Al igual que ocurre con el nombre, el valor de un nodo está en función de su tipo. Así por ejemplo cuando un nodo es del tipo *TEXT_NODE*, el valor devuelto por el método *getNodeValue* es la cadena de texto que se corresponde con dicho nodo.

Un nodo es un elemento dentro del árbol *XML*, y por lo tanto se pueden ìvisitarî otros nodos desde él: sus descendientes, sus hermanos, sus padres, etcétera. Para ello, la interfaz *Node* cuenta con los métodos *public boolean hasChildNodes(), public Node getFirstChild(), public Node getLastChild(), public NodeList getChildNodes(), public Node getParentNode(), public Node getPreviousSibling(), public Node getNextSibling(), public Document getOwnerDocument(), y public NamedNodeMap getAttributes().*

El método *hasChildNodes* indica si el nodo tiene descendientes. Si es así, serán accesibles de diversas maneras. Los métodos *getFirstChild* y *getLastChild* devuelven respectivamente el primero y el último de los nodos hijo del nodo. El método getChildNodes proporciona una lista, representada por la interfaz *NodeList*, con todos los nodos hijo. En el ejemplo que realizaremos a continuación veremos cómo es posible recorrer esa lista.

Todos los elementos de un documento XML son nodos

El método *getParentNode* se utiliza para acceder al nodo padre. Los métodos *getPreviousSibling* y *getNextSibling* facilitan los nodos hermano anterior y posterior respectivamente.

El árbol *XML* tiene siempre como raíz un nodo del tipo *DOCUMENT_NODE*. El método *getOwnerDocument* ofrece una manera rápida de llegar a ese nodo desde cualquier otro.

Finalmente, los nodos que se corresponden con etiquetas de la fuente de datos *XML* –los nodos del tipo *ELEMENT_NODE*- pueden tener a su vez descendientes ìespecialesî. Estos son los nodos que representan los atributos de las etiquetas. El método *getAttributes* devuelve una estructura, representada por la interfaz *NamedNodeMap*, que permite acceder a todos y cada uno de los atributos.

Con lo que hemos visto hasta el momento, podemos escribir de nuevo el ejemplo del capítulo anterior de forma que ahora se muestre el árbol *XML* correspondiente al archivo *noticia.xml*, además de comprobar que se trata de un documento *XML* bien formado y válido. Para ello hemos desarrollado el siguiente procedimiento:

```
public static void recorrer(Node n,
    int nivel) {
    NodeList nl = n.getChildNodes();
    System.out.println(blancos(nivel)
    +
        getNodeTypeByName(n.getNo-
    deType()) +
        "  " +
        n.getNodeName() +
        "  " +
        processValue(n.getNodeVa-
    lue()) +
        "' ");
    for(int i=0; i<nl.getLength();
    i++) {
        recorrer(nl.item(i), ni-
    vel+1);
    }
```

Recibe como parámetros un nodo del árbol *XML* y un valor numérico que se corresponde con la profundidad en que se encuentra dicho árbol. Este valor se utilizará para la salida que muestra esta aplicación.

Para el nodo actual se muestra una cadena de texto configurada de la siguiente manera:

```
1 La función blancos genera una cade-
    na de blancos acorde con el nivel
    de profundidad:
public static String blancos(int n) {
    StringBuffer sb = new StringBuf-
    fer();
    if (n>0) {
      for(int i=0; i<n; i++) {
          sb.append("——");
      }
      sb.append("  ");
    }
    return sb.toString();
}
```

- Se utiliza la función *getNodeTypeByName* descrita anteriormente para ob-

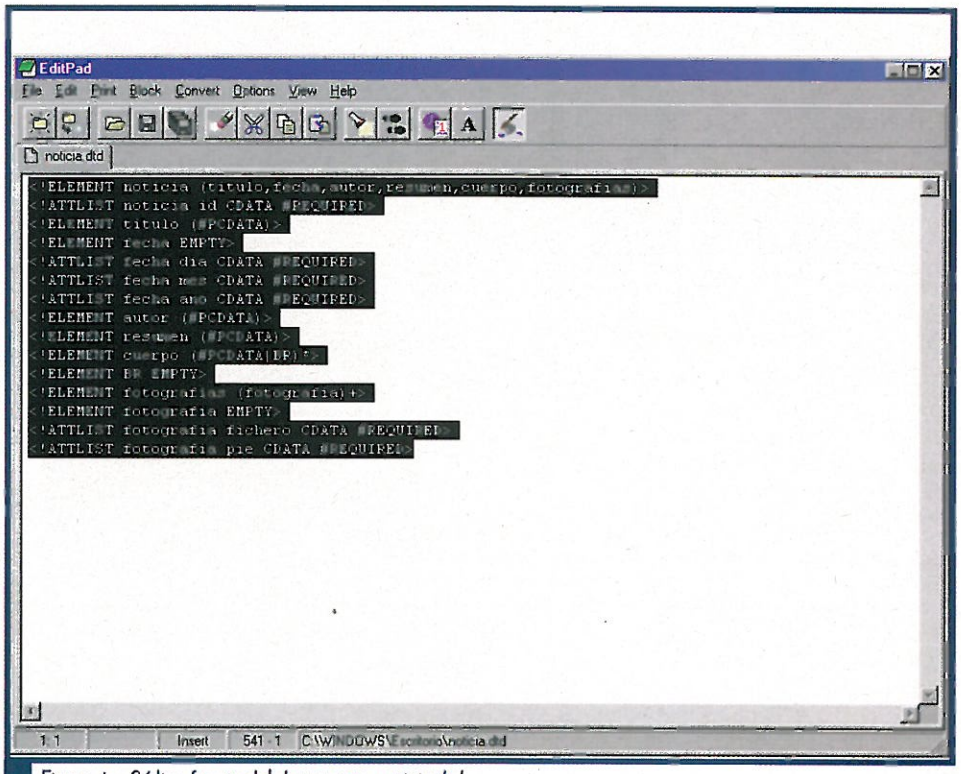


Figura 4. - Código fuente del documento *noticia.dtd.*

tener el nombre del tipo del nodo a partir del valor numérico devuelto por el método *getNodeType*.

- Se muestra el nombre del nodo facilitado por *getNodeName*.
- Y por último, se muestra el valor del nodo. Éste se procesa previamente para acortarlo, de manera que la salida por la pantalla sea lo más legible posible.

Los nodos del tipo ELEMENT_NODE son las etiquetas XML

El método *getChildNodes* devuelve una lista con todos los descendientes del nodo actual. Para cada uno de ellos se llama recursivamente a la misma función, con lo que así se recorre todo el árbol. Además, se incrementa el valor del parámetro *nivel* para indicar la profundidad de los nodos subsiguientes.

Para mostrar todo el árbol *XML* del documento *noticia.xml* solamente hay que modificar el código fuente original del la función *main*:

```
...
doc = docBuilder.parse(new File("NOTI-
    CIA.XML"));
doc.getDocumentElement().normalize();
recorrer(doc, 0);
...
```

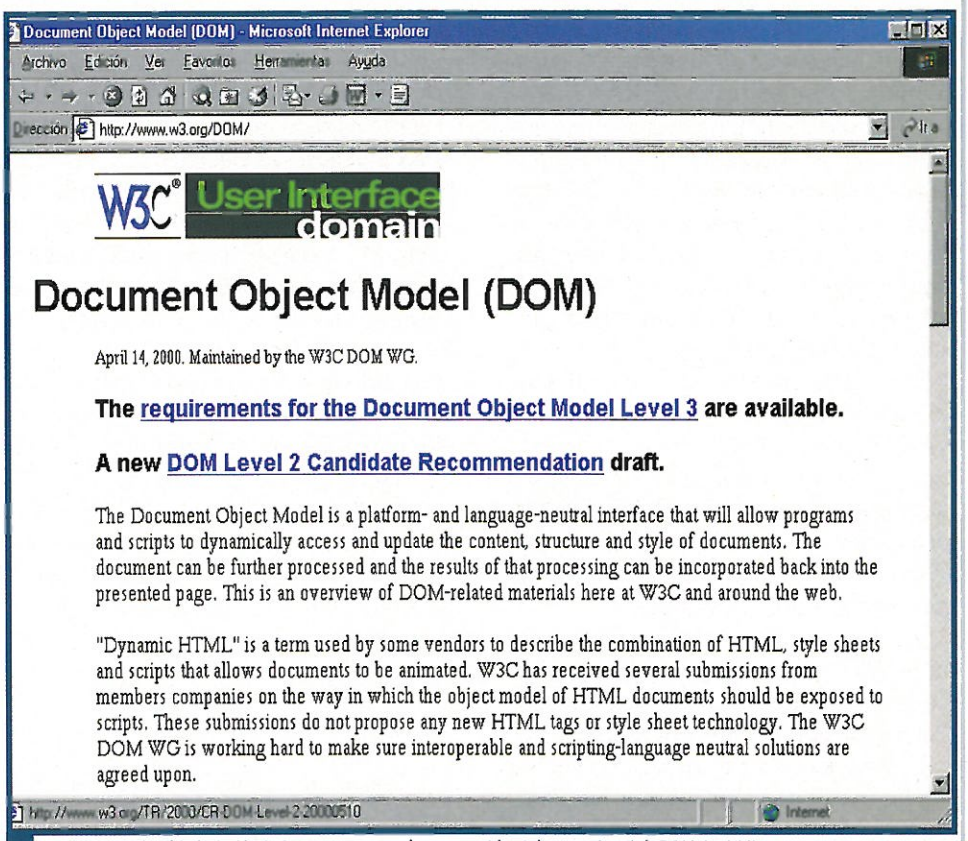


Figura 6. - *World Wide Web Consortium* es el responsible del estándar del *DOM* de *XML*.

La Figura 5 muestra el resultado obtenido al ejecutar la aplicación. La raíz del árbol *XML* es un nodo del tipo *DOCUMENT_NODE*. Su nombre es la cadena *#document*, y no tiene ningún valor asociado, pues el metodo *getNodeValue* devuelve *null*. Este nodo tiene dos descendientes: un nodo del tipo *DOCUMENT_TYPE_NODE*, que representa la instrucción *<!DOCTYPE noticia SYSTEM "noticia.dtd">* mediante la que se asocia el documento *DTD*, y un nodo del tipo *ELEMENT_NODE* que se corresponde la etiqueta *noticia*. A partir de aquí, se suceden los diferentes nodos que conforman la estructura del documento *XML*.

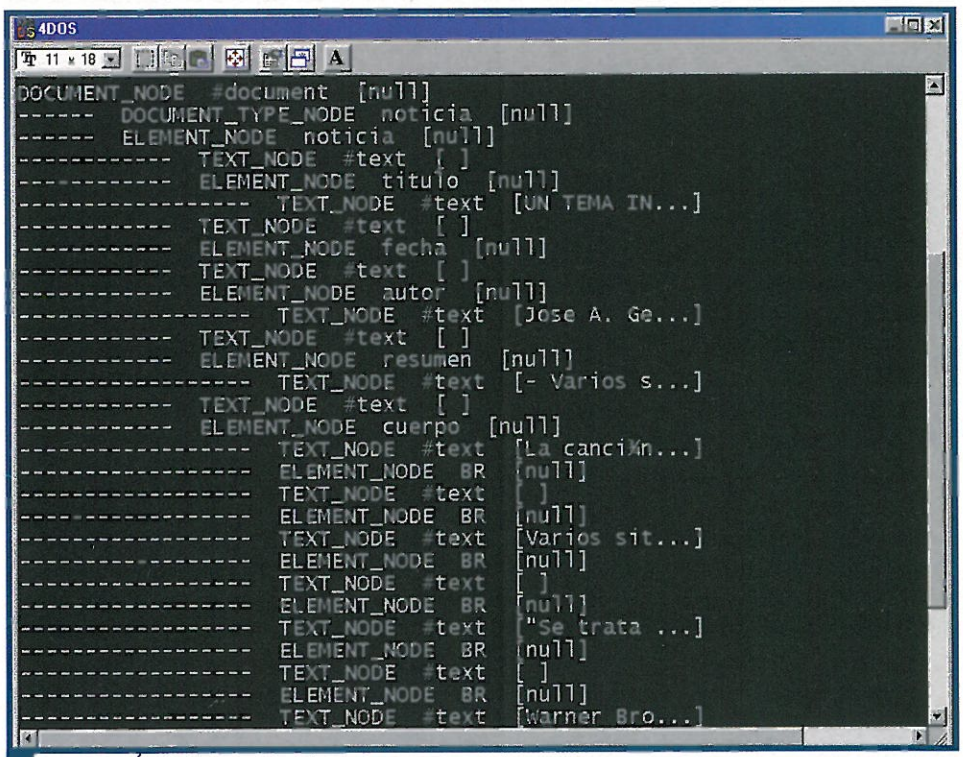


Figura 5. - Árbol *XML* asociado al documento noticia.xml.

Nótese que aparecen con frecuencia nodos del tipo *TEXT_NODE*, que en principio, no forman parte de la noticia tal y como se ha estructurado. Estos nodos corresponden a los espacios en blanco (blancos, tabuladores o saltos de línea) existentes entre las etiquetas *XML*. Para el analizador constructor del *DOM*, todo el contenido de un archivo *.xml* ha de conformar el árbol *XML*.

En los siguientes capítulos seguiremos profundizando en estos aspectos del *DOM*, y estudiaremos detenidamente las particularidades de cada uno de los tipos de nodos.

SUSCRÍBETE

A

LA PRIMERA REVISTA DE PROGRAMACIÓN EN CASTELLANO

SÓLO PROGRAMADORES

QUE NO TE LÍEN CON EL <CÓDIGO>

- NOTICIAS
- MULTIMEDIA
- TEORÍA
- INTERNET
- DESARROLLO DE APLICACIONES
- COMUNICACIONES
- HERRAMIENTAS
- PROGRAMACIÓN
- ÚLTIMAS TENDENCIAS
- Y MUCHO MÁS

BOLETÍN DE SUSCRIPCIÓN

Rellene o fotocopie el cupón y envíelo a REVISTAS PROFESIONALES, S.L. (Revista Sólo Programadores).
C/ San Sotero, 5. 1ª Planta. 28037 Madrid. Tlf: 91 304 87 64. Fax: 91 327 13 03

Quiero suscribirme a la revista SÓLO PROGRAMADORES desde el Número .....
y beneficiarme de las condiciones de esta magnífica promoción:

❑ **SUSCRIPCIÓN ANUAL**
11 NÚMEROS + 11 CD-ROMs
AL PRECIO DE
8.580 ptas. / 51,57€

Precio de suscripción para el extranjero:
11 NÚMEROS + 11 CD-ROMs
AL PRECIO DE
11.800 Ptas.

FORMAS DE PAGO:

❑ Giro postal a nombre de REVISTAS PROFESIONALES, S.L
❑ Transferencia al Banco Popular Español. C/ Valdecanillas, 41.
Nº c/c: 0075/1040/43/ 0600047439
❑ Talón bancario a nombre de REVISTAS PROFESIONALES, S.L
❑ Domiciliación bancaria
❑ Contra reembolso

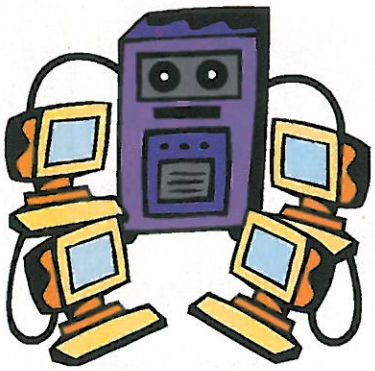
Soy antiguo suscriptor
❑ Sí ❑ No

PARA ENVÍOS AL EXTRANJERO SÓLO SE ADMITIRÁN LAS SIGUIENTES FORMAS DE PAGO:
❑ Giro postal a nombre de REVISTAS PROFESIONALES, S.L
❑ Eurocheque conformado con un banco español a nombre de REVISTAS PROFESIONALES, S.L
❑ Por VISA ____/____/____/____
Caducidad: __/__

NOMBRE Y APELLIDOS:.....
EDAD:..... PROFESIÓN: .....
TFNO: ..... DOMICILIO: .....
CIUDAD:..... C.P: .....PROVINCIA:.....

Datos de domiciliación:
Banco.....
Domicilio .....
Nº de Cuenta.....
Titular.....
Fecha .....
FIRMA

Promoción válida hasta agotar existencias



JSP (I)

Todas las ventajas de la programación Java

Adolfo Aladro García. *Analista Programador.*

Las tecnologías de soporte a las aplicaciones Web están adquiriendo un auge acorde con el desarrollo de *Internet*. *JSP (Java Server Pages)*, parte de la plataforma Java, es la respuesta de *Sun* para crear aplicaciones *Web* potentes y versátiles.

CARACTERÍSTICAS DE LAS PAGINAS JSP

Las páginas *JSP* llevan *Java* al servidor, de la misma manera que hacen los *servlet*s, si bien las primeras crean un entorno de desarrollo más sencillo y versátil cuando hay que crear páginas *HTML* dinámicas y sofisticadas.

La plataforma *Java* es conocida por su potencial para crear aplicaciones de cliente que interaccionan con la *Red*. Pero *Java* también es un poderoso recurso para desarrollar código que se ejecuta en el servidor. Las páginas *JSP* aprovechan todas las ventajas que proporciona *Java* como lenguaje de programación. Además, esta tecnología forma parte de la plataforma *J2EE (Java 2 Enterprise Edition)*. Esto facilita la creación de soluciones integrales que abarcan todo tipo de aplicaciones.

En la actualidad el tiempo de desarrollo es uno de los requisitos más importantes a la hora de crear aplicacio- En este sentido los *servlets* suficientes cuando se crean es *Web* en el que hay un número considerable de páginas con un diseño complejo.

SINTAXIS

En realidad, las páginas *JSP* no son más que *servlets*. Sin embargo, su utilización facilita los procesos de desarrollo, ayuda a separar la presentación de la lógica de las aplicaciones, y por lo tanto facilita que las aplicaciones se creen con mayor rapidez y fiabilidad. La mayor parte de los servidores que soportan páginas *JSP*, o de los *plug-ins* que proporcionan esta funcionalidad para aquellos otros servidores que no pueden ejecutar páginas *JSP* directamente, procesan las páginas *JSP* proporcionando *servlets*. Esta es una tarea en la que suceden de manera transparente al programador.

Desde el punto de vista del programador, las páginas *JSP* son páginas *HTML* en las que se intercalan trozos de código que se ejecuta en el servidor. Para ello se utilizan etiquetas especiales. La mayor parte de ellas comienzan por <% y terminan con %>. Por ejemplo:

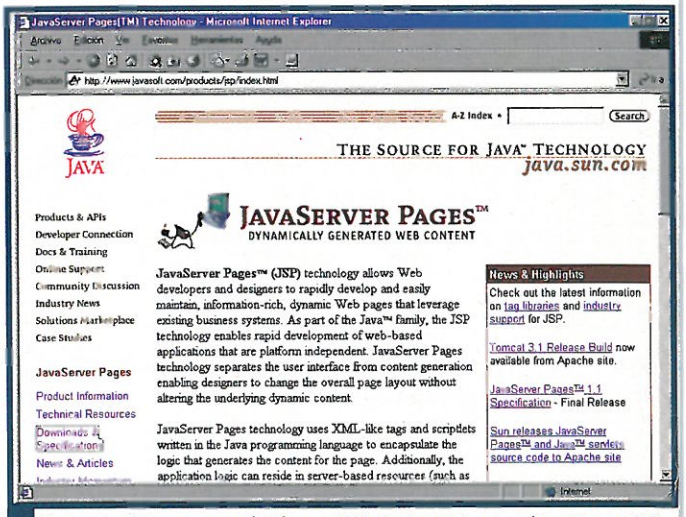


Figura 1. - Página principal del sitio *Web* de *Sun* dedicado a las páginas *JSP*.

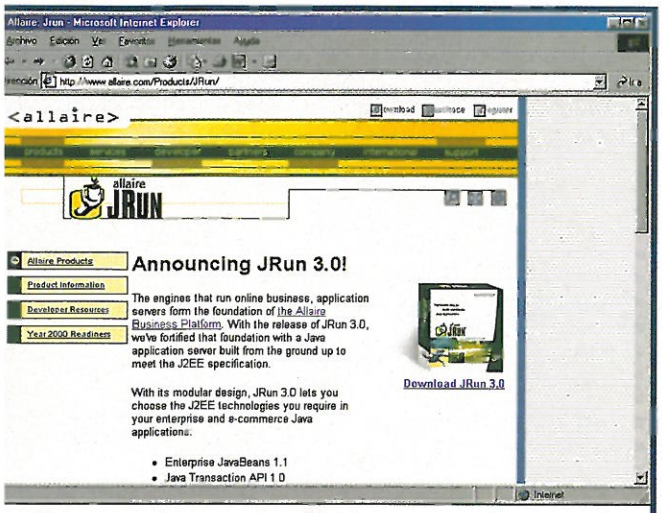


Figura 2. - *Allaire JRun* es un *plug-in* sirve para ejecutar *servlets* y páginas *JSP* es un gran número de servidores *Web*.

```
HTML>
HEAD>
TITLE>Prueba</TITLE>
LINK REL="STYLESHEET" TYPE="text/css"
    HREF="estilo.css">
HEAD>
BODY BACKGROUND="FondoSPP.gif">
y es...<%= new java.util.Date(); %>
BODY>
HTML>
```

Esta página *JSP* muestra un mensaje n la fecha del día actual. La expresión %= ··· %> se utiliza para mostrar en pantalla el resultado de evaluar el có- go Java contenido en la misma.

s páginas JSP se convierten servlets

Las páginas *JSP* tienen la extensión *p*, y se deben colocar en el lugar indi- do, dependiendo del tipo de servidor aplicaciones que utilicemos. Aunque aspecto que presenta una página *JSP* el de un documento *HTML*, la mayor rte de los servidores de aplicaciones raducen" las páginas *JSP* a *servlets*. La imera vez que se solicita una página *P* se convierte en un *servlet*, y enton- s se compila. La clase correspondien- es cargada por el servidor. Sucesivas madas a la página *JSP* son atendidas r *threads* de ese *servlet*, que ya está rgado en memoria. Así se incrementa rendimiento de la aplicación *Web*.

Teniendo en cuenta lo anterior, las ginas *JSP* pueden ser estudiadas desde la perspectiva del *servlet* en el que se convierten. Por esta razón, conviene recordar la estructura general de un *servlet*.

Los *servlets* tienen un ciclo de vida predeterminado. Está compuesto por tres fases: inicio, procesamiento de las peticiones, y terminación. Generalmente un *servlet* se inicia la primera vez que recibe una petición. En ese momento el servidor de aplicaciones crea una concreción y llama al método *init*. Este método es invocado una sola vez. Por lo tanto, todas las tareas que tengan un propósito global, deben ser llevadas a cabo en el código correspondiente a este método.

El servidor de aplicaciones crea un objeto *Request* (petición) y un objeto *Response* (respuesta) por cada petición. Éstos son pasados como parámetros al método *service* del *servlet*. El objeto *Request* proporcionan toda la información relativa a la petición. El objeto *Response* permite elaborar la respuesta que será enviada al navegador del usuario. Las peticiones se atienden concurrentemente.

Finamente, el servidor de aplicaciones llama al método *destroy* cuando ya no necesita a ese *servlet*, o bien éste va a ser cargado de nuevo. Este método se encarga de liberar todos los recursos que el *servlet* consume.

En la sintaxis de las páginas *JSP* podemos distinguir los siguientes tipos de elementos o constructores:

- Elementos de código

Permiten crear el código *Java* que formará más tarde parte del *servlet* que se cree a partir de la página *JSP*.

- Directivas

Permiten controlar la configuración general del *servlet* que se cree a partir de la página JSP.

- Acciones

Las acciones se corresponden con objetos que pueden ser utilizados para llevar a cabo determinadas funcionalidades.

- Objetos

Las páginas *JSP* cuentan con una serie de objetos por defecto. En realidad, no son más que los objetos que se manejan en un *servlet*.

La Tabla 1 muestra un resumen de los elementos más importantes de la sintaxis de las páginas *JSP*.

Ahora nos encontramos en disposición de crear nuestras primeras páginas *JSP*. Antes de ello vamos a ver cuál es el entorno de desarrollo necesario y cómo debemos configurarlo.

ENTORNO DE DESARROLLO

Para desarrollar una aplicación *Web* con páginas *JSP* necesitamos un servidor de aplicaciones capaz de ejecutar este tipo de componentes. Existen multitud de configuraciones, especialmente si consideramos entornos profesionales. En el caso que nos ocupa vamos a proponer un entorno de desarrollo muy sencillo que sirva únicamente a nuestro propósito, que no es otro que el de un primer contacto con esta tecnología. En cualquier caso, cualquier otra configuración no implicará ningún cambio sustancial en lo aquí expuesto, ya que nos ajustaremos al estándar de *Sun*.

Para crear un entorno cliente-servidor en nuestro *PC* es preciso tener un servidor *Web*. El servidor *PWS (Personal Web Server)* de *Microsoft* es más que

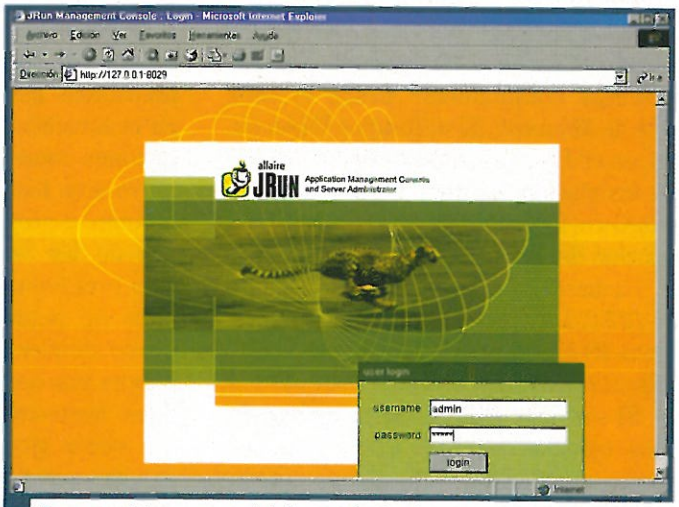


Figura 3. - Página principal de la consola de administración de *JRun*.

TABLA 1. Resumen de los elementos más importantes de la sintaxis de las páginas JSP.

ELEMENTOS DE CÓDIGO		
Expresión	<%= expresión %>	La expresión se evalúa y el resultado se muestra por la salida.
Código	<% código %>	Este código se ejecuta en el método service del servlet correspondiente.
Declaración	<%! declaración %>	Este código se inserta en el cuerpo de la clase del servlet correspondiente, fuera del método service.
Comentario	<%— comentario —%>	Los comentarios se ignoran cuando la página se convierte en un servlet.
DIRECTIVAS		
Directiva de página	<%@ page atributo="valor" %>	Indicaciones que se dan al motor de servlets acerca de los parámetros generales de ejecución para esa página. Los valores posibles son: • import="package.class" • contentType="MIME-Type" • isThreadSafe="true|false" • info="mensaje" • session="true|false" • errorPage="URL" • buffer="tamañoKb|none" • isErrorPage="true|false" • autoflush="true|false" • lenguaje="Java" • extends="package.class"
Directiva de inclusión	<%@ include file="ruta" %>	Se incluye el fichero indicado dentro de la página JSP antes de que esta vaya a ser procesada.
ACCIONES		
Acción jsp:include	<jsp:include page="URL relativa" flush="true"/>	La acción jsp:include se usa para incluir el resultado de ejecutar una página dentro de otra.
Acción jsp:forward	<jsp:forward page=" URL relativa"/>	La acción jsp:forward se usa para reenviar la petición a otra página JSP.
OBJETOS		
Objeto de aplicación	application	El objeto ServletContext para esa página JSP.
Objeto de configuración	config	El objeto ServletConfig para esa página JSP.
Objeto de excepción	exception	La excepción no capturada.
Objeto de salida	out	Este objeto sirve para escribir en el canal de salida que da lugar a la página que se ve el usuario en el navegador.
Objeto de petición	request	El objeto HttpServletRequest para esa página JSP.
Objeto de respuesta	response	El objeto HttpServletResponse para esa página JSP.
Objeto de sesión	session	El objeto sesión creado para ese usuario.

suficiente para comenzar. Si utilizamos *Windows 95/NT*, el servidor *PWS* puede instalarse desde el *Option Pack*. Éste se descarga gratuitamente desde el sitio *Web* de *Microsoft*. Si utilizamos *Windows 98*, en el directorio *\add-ons\pws* del *CD* de instalación, se encuentra el archivo *setup.exe*. Ejecutándolo, comienza la instalación del servidor.

La dirección *IP* de nuestro ordenador es *127.0.0.1* y podemos acceder a las páginas *Web* a través de la dirección URL *http://127.0.0.1/*.

El servidor *PWS* no está preparado para ejecutar *servlets*. Como complemento al servidor Web, hemos optado por *JRun* de *Allaire*. Recientemente ha lanzado la versión *3.0*, que cuenta con un gran número de características nuevas. Lo que empezó siendo un pequeño y modesto *plug-in* para poder ejecutar *servlets* en servidores que no estaban preparados para ello, se ha convertido en la actualidad en un servidor de aplicaciones moderno, potente y versátil. La versión llamada *Developer Edition* es gratuita, y proporciona todas las funciones de las versiones comerciales, si bien está limitado a tres el número máximo de peticiones concurrentes que puede atender. No obstante, esa versión resulta una herramienta excelente para fines ilustrativos, ya que proporciona una visión global y completa de lo que es una servidor de aplicaciones moderno basado en la plataforma *Java* que cumple con la mayor parte de los estándares. La última versión de *JRun* puede descargarse del sitio *Web* de *Allaire (www.allaire.com/Products/JRun/)*.

CONSOLA DE ADMINISTRACIÓN

La instalación de *JRun* es bastante sencilla. Ahora bien, su utilización no lo es tanto, ya que dispone de un gran número de opciones. Por ello, es aconsejable, al menos al principio, seguir los pasos que señalaremos. A medida que nos familiaricemos con la aplicación, podremos pasar a utilizar todas las funcionalidades disponibles.

La Figura 3 muestra la página principal de la consola de administración de *JRun*. Antes de acceder a ella, hay que arrancar el servidor *Web*, el servidor *JRun Default* y el servidor *JRun Admin*. Si durante la fase de instalación

Direcciones

JavaServer Pages (TM) Technology
www.javasoft.com/products/jsp/index.html

Allaire Jrun
www.allaire.com/Products/JRun/

JSP JavaServer Pages Archive
jsp-interest.com/

Web Development with JavaServer Pages
www.burridge.net/jsp/

emos elegido instalar *JRun* como un ervicio, los servidores de *JRun* esta- án listos para funcionar nada más ini- iar el ordenador.

a duración de un servlet marca a de las páginas JSP

En otro caso, será preciso arrancar- os manualmente yendo a *Inicio > Pro- ramas > Jrun 3.0* y haciendo clic en las pciones oportunas. Nada más hacer es- o se mostrarán los iconos correspon- ientes en el área de iconos de notifica- ión, junto al reloj de *Windows*.

La dirección URL que nos lleva a la ágina principal de la consola de admi- istración de JRun es *http://127.0.0.1 3029/*.

El puerto (*8029*) es uno de los datos ue se piden durante la instalación, por lo que puede ser diferente para cada usuario.

Por último, la página principal solicita usuario y contraseña antes de entrar en la consola de administración. Estos datos también se requieren durante la instalación.

La consola de administración de *JRun* nos permite configurar todos los parámetros que requieren nuestras aplicaciones *Web*, en lo que concierne estrictamente a cada uno de las aplicaciones así como en lo que afecta al funcionamiento global del servidor de aplicaciones.

CREAR UNA APLICACIÓN WEB

Antes de crear una aplicación *Web*, es conveniente haber realizado la conexión adecuada entre el servidor *JRun* y el servidor *Web*, que en el caso que nos ocupa es el servidor *PWS*. Esta operación puede llevarse a acabo durante el proceso de instalación, o bien más tarde, desde la consola de administración siguiendo unos pasos sencillos.

Para crear una aplicación *Web* tenemos que ir a la consola de administración y seleccionar en el menú de la izquierda la opción *JRun Default Server*. Entonces se desplegará un árbol de opciones. Elegimos la que se denomina *Web Applications,* y entonces en el panel de la derecha hacemos clic en la opción *Create an Application*. La aplicación *Web* que vamos a utilizar para albergar nuestras páginas *JSP* se ha configurado de la siguiente manera:

- *JRun Server Name: Jrun Default Server*
- *Application Name: prueba*
- *Application Host: All Hosts*
- Application URL: */prueba*
- Application Root Dir: *C:\WINDOWS\Escritorio\prueba*

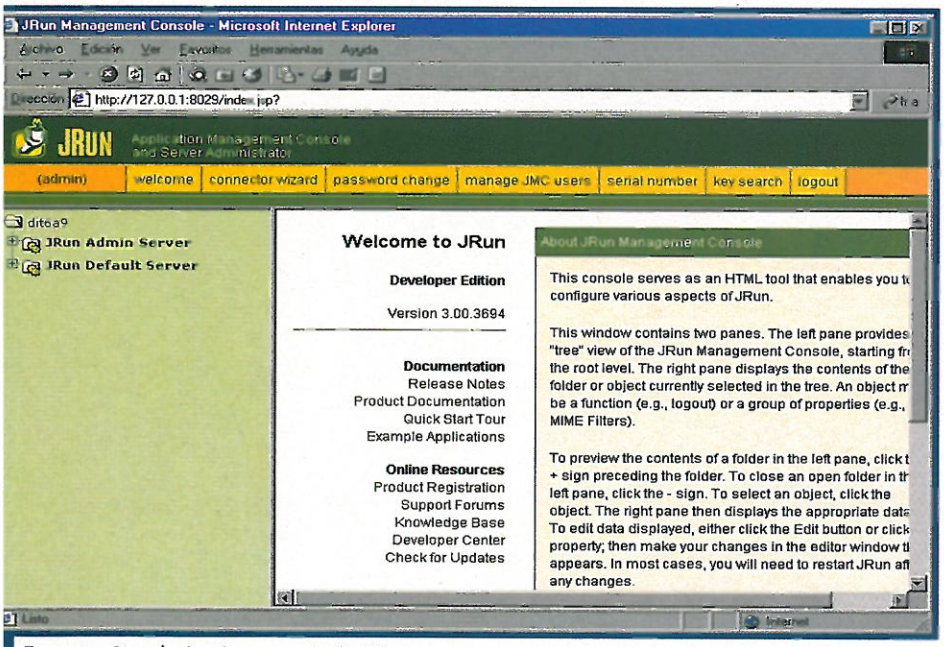


Figura 4. - Consola de administración de *JRun*.

JRun permite crear aplicaciones Web basadas en servlets y páginas JSP

Sin entrar en detalles por el momento, lo anterior significa únicamente que en el directorio *C:\WINDOWS\Escritorio\prueba* situaremos las páginas *JSP* que desarrollemos, y que se podrá acceder a éstas por la dirección URL *http://127.0.0.1/prueba/*

Es preciso reiniciar los servidores para que la aplicación *Web* empiece a funcionar correctamente.

NUESTRA PRIMERA PÁGINA JSP

La Figura 6 muestra el resultado de llamar a nuestra primera página *JSP*. El código fuente de la misma es:

```
<%@ page import="java.util.*" %>

<HTML>
<HEAD>
<TITLE>S&oacute;lo Programadores - JSP (I)</TITLE>
<LINK REL="STYLESHEET" TYPE="text/css"
    HREF="estilo.css">
</HEAD>
<BODY BACKGROUND="FondoSPP.gif">
<%@ include file = "cabecera.jsp" %>
Hoy es...<%= new Date(); %>
<%@ include file = "pie.jsp" %>
</BODY>
</HTML>
```

Como se puede observar, la página *JSP* no es más que una página *HTML* en la que se intercalan segmentos de código que es interpretado por el servidor de aplicaciones.

En la primera línea utilizamos una directiva de página mediante la que se indica que se van a utilizar las clases paquete *java.util*.

La página JSP utiliza una cabecera y un pie que se insertan en la página me-

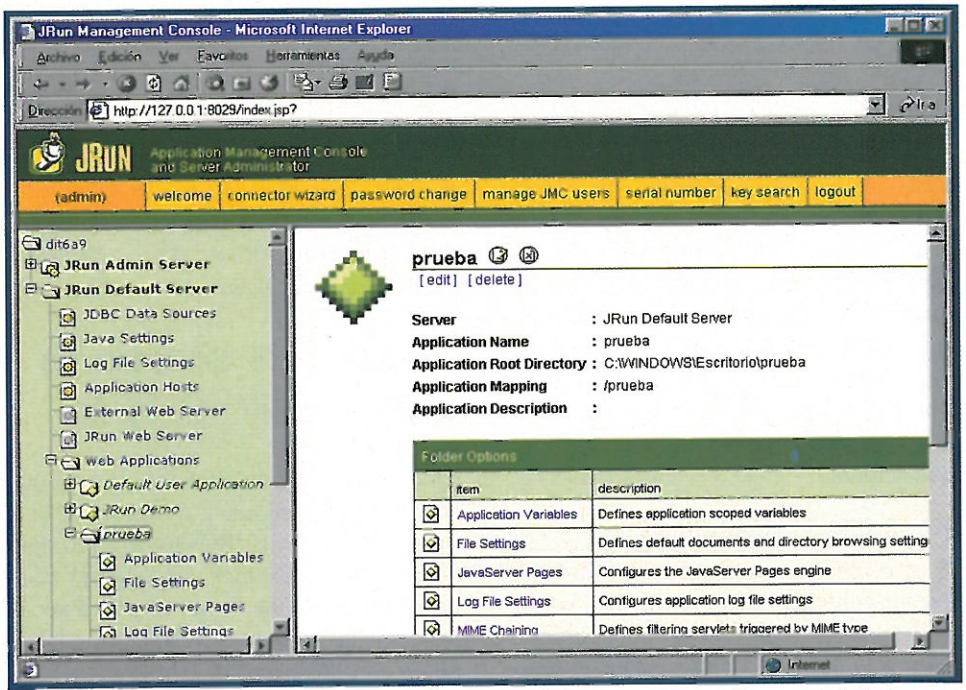


Figura 5. - Configuración de la aplicación *Web* llamada "prueba" y que estará accesible a través de la dirección *URL http://127.0.0.1/prueba.*

diante directivas de inclusión. La primera se encuentra inmediatamente después de la etiqueta *<BODY>*, y la segunda, justo antes del cierre de la etiqueta anterior, *</BODY>*.

En el cuerpo de la página se utiliza la expresión *new Date()* y se muestra el resultado. Para ello se usa la etiqueta *<%= ··· =>*

NUESTRA SEGUNDA PÁGINA JSP

Ahora que acabamos de hacer un sencillo ejemplo, es el momento de complicarlo un poco. La página que sigue, muestra igualmente la fecha actual, aunque en este caso, ésta aparece con un formato diseñado por nosotros mismos mediante el código *Java*.

```
<%@ page import="java.util.*" %>
<%
String[] meses = {"Enero", "Febrero", "Marzo",
      "Abril", "Mayo", "Junio", "Julio", "Agosto",
      "Septiembre", "Octubre", "Noviembre", "Di-
      ciembre"};
String[] dias_semana = {"Domingo", "Lunes", "Mar-
      tes", "Mi&eacute;rcoles", "Jueves", "Viernes",
      "S&aacute;bado"};
Calendar c = Calendar.getInstance();
int dia_semana = c.get(Calendar.DAY_OF_WEEK);
int dia = c.get(Calendar.DAY_OF_MONTH);
int mes = c.get(Calendar.MONTH);
int ano = c.get(Calendar.YEAR);
%>
<HTML>
<HEAD>
<TITLE>S&oacute;lo Programadores - JSP (I)</TI-
      TLE>
<LINK REL="STYLESHEET" TYPE="text/css"
      HREF="estilo.css">
</HEAD>
<BODY BACKGROUND="FondoSPP.gif">
<%@ include file = "cabecera.jsp" %>
Hoy es...<% out.println(dias_semana[dia_semana-1]
      + ", " + dia + " de " + meses[mes] + " de "
      + ano); %>
<%@ include file = "pie.jsp" %>
</BODY>
</HTML>
```

La clase *Calendar* sirve para representar fechas. Se utilizan dos *arrays* de cadenas de texto (*String*), *meses*, y *dias_semana*, que guardan los nombres de los meses del año y los nombres de los días de la semana. El método *get* de la clase *Calendar* devuelve el valor correspondiente a la constante que recibe como parámetro. Así, se van obteniendo el día, el número del día, el mes y el año de la fecha actual.

Las directivas controlan aspectos globales de la ejecución

En el ejemplo anterior, se utilizaba el elemento *<%= ··· =>* para evaluar una expresión *Java* y mostrar el resultado por la salida. En este caso, se utiliza el método *println* del objeto *out*. El resultado es igual: traslada una cadena de texto a la salida.

Hasta aquí este primer acercamiento a *JSP*. En posteriores artículos iremos aprendiendo más de esta útil herramienta que nos ayudará a mejorar los procesos de desarrollo.

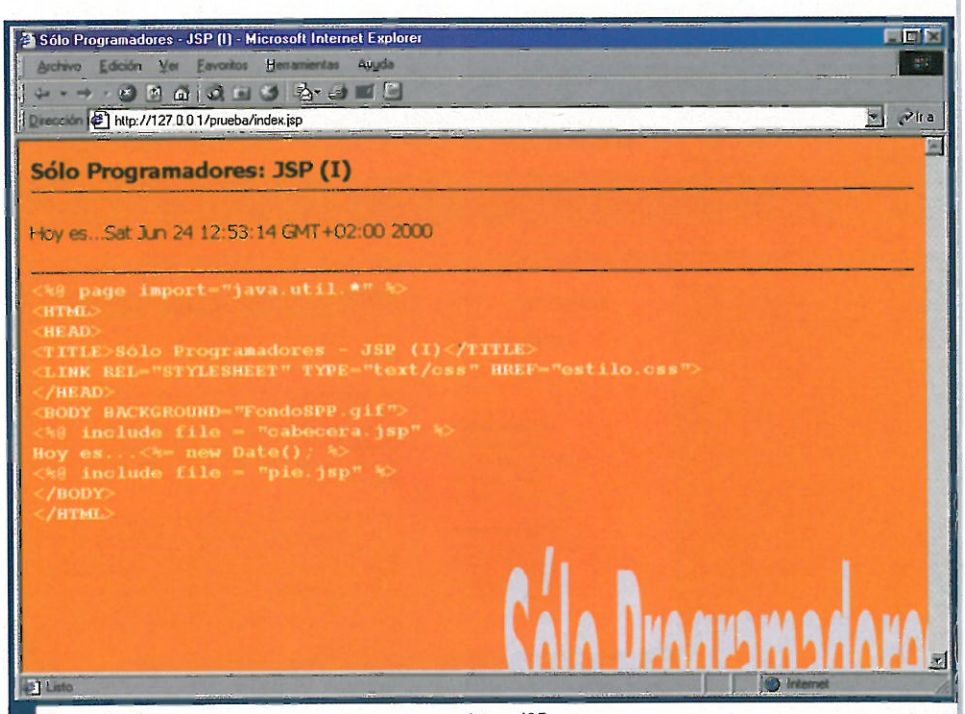


Figura 6. - Aspecto que presenta nuestra primera página *JSP*.

Ahora puedes solucionarlo

Por fin los genuinos archivadores de SÓLO PROGRAMADORES

Podrás hacerte con ellos al precio de:

- Archivador de revistas: 900 Ptas.* (5,41€)
- Archivador de CD's: 700 Ptas.* (4,20€)
- Especial ahorro. Los dos archivadores por sólo 1.400 Ptas.* (8,41€)

Llama antes de que se agoten al teléfono 91 304 87 64, o escribe a la siguiente dirección:

BOLETÍN DE PEDIDO

Rellene o fotocopie el cupón y envíelo a REVISTAS PROFESIONALES, S.L.
(Revista Sólo Programadores Linux). C/ San Sotero, 5. 1ª Planta. 28037 Madrid
Tlf: 91 304 87 64. Fax: 91 327 13 03

Deseo que me envíen .....

.....

NOMBRE Y APELLIDOS: .....

EDAD: ..... PROFESIÓN: ..... TFNO: .....
DOMICILIO: .....
CIUDAD: .....
PROVINCIA: ..... C.P: .....

Gastos de envío: 700 Ptas.

FORMAS DE PAGO:

- ❑ Giro postal a nombre de REVISTAS PROFESIONALES, S.L.
- ❑ Talón bancario a nombre de REVISTAS PROFESIONALES, S.L.
- ❑ Domiciliación bancaria
- ❑ Contra reembolso

DATOS DE DOMICILIACIÓN:

Banco ..... FIRMA

Domicilio .....

Nº de Cuenta | | | .....

Titular .....

Fecha .....

Optimación de aplicaciones Java

Javier Sanz Alamillo.
Ingeniero de Software.

En este artículo comprobaremos que se puede mejorar considerablemente el rendimiento de las aplicaciones, sin necesidad de recurrir a programas comerciales.

Java nació como el lenguaje de programación preferido por los desarrolladores para la creación de aplicaciones para *Internet*, por su alto grado de portabilidad, integración con *HTML* y la sencillez de uso, mayor que la de otros lenguajes como *C++* o *ADA* para proyectos orientados a objetos.

Sin embargo, ha existido una cierta reticencia al desarrollo de aplicaciones y servidores de alto rendimiento en *Java*, pues en muchas aplicaciones, el rendimiento conseguido era bastante menor que el obtenido con lenguajes como *C++*, lo que descartaba a *Java* para la construcción de desarrollos de alto rendimiento.

Con el tiempo, cuando se mejoró en especificación y rendimiento, se empezaron a utilizar los famosos *applets* en páginas *webs*, y todo ha ido creciendo y mejorándose hasta tal punto, que *Java* es el lenguaje por excelencia para el desarrollo de cualquier tipo de proyectos. Actualmente, las aplicaciones distribuidas escritas en *Java* tienen una estabilidad y rendimiento muy próximo a la de C++, e incluso en algunos tipos de aplicaciones, el resultado obtenido es claramente superior.

Pero no todo marcha tan bien como se espera. Los desarrolladores encuentran *Java* poco estable en algunas aplicaciones que exigen muchos recursos, y su rendimiento se degrada incluso con el uso de un *JIT* (*Just In Time Compiler*). Esta degradación se debe a una cierta falta de capacidad para realizar optimaciones de calidad a los programas, pues esta característica de *Java* no está suficientemente documentada o accesible, y lo que es peor, en los proyectos se llega fuera de fecha y se olvida o descarta este proceso. Se asume que *Java* hace por nosotros todo lo que se necesita, como un recolector de basura.

Empresas importantes, grandes desarrolladores de *software* y multinacionales de telecomunicaciones, se han volcado en exigir a sus proyectos un mínimo de optimaciones y un control y verificación de la ejecución mediante herramientas denominadas *profilers* y *memory checkers*.

El rendimiento puede mejorar con sencillas optimizaciones

Donald Knuth, el gran pionero en el desarrollo de algoritmos y de las bases algorítmicas de la informática, afirmaba que "La optimización prematura es la madre de todos los males",o dicho de otra forma nunca se debe realizar una optimación cuando aún se está desarrollando:

- A no ser que el programa esté funcionando y se quiera garantizar un tiempo de ejecución correcto, no realice optimación alguna, ya que puede producir nuevos errores.

```
public void f(int i)
{
//    0    0:iconst_0
//    1    1:istore_2
//    2    2:goto          15
//    3    5:getstatic     #2   <Field PrintStream System.out>
//    4    8:iload_2
//    5    9:invokevirtual #3   <Method void PrintStream.println(int)
//    6   12:iinc          2  1
//    7   15:iload_2
//    8   16:iload_1
//    9   17:icmplt        5
//   10   20:return
}
```

Figura 1. - *Bytecode* bucle 1.

La optimación hará más difícil el mantenimiento del código... ¿hasta qué punto interesa entonces?

Un incremento en la velocidad de ejecución puede demandar más recursos, no siempre disponibles.

El código optimado para una plataforma determinada puede ejecutarse peor en otra diferente.

Muchas veces ocurre que el modesto incremento de rendimiento no compensa el tiempo empleado en la optimación.

TÉCNICAS GENERALES

n lenguajes compilados, como *C++*, el tiempo de acceso a las variables es nilar, ya que se predetermina, o como ucho se accede mediante un nivel de *dirección*. En cambio, en *Java* ese mpo no es tan estable, depende del ti- de variables accedidas y de su uso. r ejemplo, las variables de *stack*(loca- en los métodos) son más rápidas, ya e requieren menos *indirección* que as, como las variables de concreción. Por ejemplo, un buen truco para una sible optimación en aplicaciones con chos datos, como operaciones mate- íticas, es copiar las concreciones de riables en variables locales, y recupe- ; al fin del proceso, el valor obtenido las locales sobre las primeras. La ra- n de esta mejora de tiempo, es que no y muchos niveles de indirección, y isten unos códigos de operación espe- les para la máquina virtual que ayu- n a mejorar el rendimiento. Una téc- a íntimamente ligada a ésta, se utiliza bucles, y es la modificación de los smos cuando es posible que la condi- n de fin se exprese mediante opera- nes con *0*, como *>0, >=0,<=0,* et- tera. Por ejemplo, un simple cambio el bucle *for (int i=0: ++i<=limit; Nada*/) //bucle 1* por *for (int i=limit; i>=0 /*Nada*/) // bucle 2,* mejora el ndimiento un 20% aproximadamente no se usa *JIT,* y de un 5% si se utiliza.

NCRONIZACIÓN

n *Java*, al igual que en otros lenguajes de programación, se puede determi- r que ciertos métodos o bloques de digo sean sincronizados. Esto signifi- que múltiples *threads* no pueden ac- der simultáneamente a ellos. Para o, la máquina virtual asigna un *moni-* a cada uno de los objetos de un método o bloque sincronizado. Para que un *thread* pueda ejecutar ese método, debe tomar el bloqueo del mismo.

```
public void f(int i)
{
//    0    0:iload_1
//    1    1:istore_2
//    2    2:goto          12
//    3    5:getstatic     #2   <Field PrintStream System.out>
//    4    8:iload_2
//    5    9:invokevirtual #3   <Method void PrintStream.println(int)>
//    6   12:iinc          2  -1
//    7   15:iload_2
//    8   16:ifgt          5
//    9   19:return
}
```

Figura 2. - *Bytecode* bucle 2.

Tras esta explicación, el lector se ha percatado de que existe un proceso interno para poder ejecutar un método u bloque sincronizado, que llevará su tiempo. Tras comprobarse en varios estudios la realidad dice que el uso de métodos sincronizados puede reducir el rendimiento hasta seis veces, incluso existiendo un único *thread* de ejecución.

Los memory leaks son difíciles de corregir

Aunque en las últimas máquinas virtuales (sobre todo *Java 2*) parece que esta degradación del rendimiento ha mejorado en un par de puntos, aún sigue pareciendo demasiado costoso. Los clásicos ejemplos de reducción de rendimiento han sido la clase *Vector* y las clases de *I/O* como *DataInputStream*. En la clase *Vector*, todos los métodos están sincronizados y en la mayoría de las aplicaciones existe un único *thread* de ejecución, por lo que se está perdiendo rendimiento. Este error se ha corregido de la forma más sencilla: En *Java 2*, la clase *Vector* no está sincronizada. Que lo recuerden algunos desarrolladores que pasan aplicaciones de *Java 1.1 a Java 2* y notan cómo algo empieza a fallar.

Un problema similar, existe con la *I/O*, donde la mayoría de los métodos están sincronizados. Hay un pequeño truco para la lectura de ficheros y similares. Consiste en re-escribir toda la clase que se vaya a utilizar, eliminando la palabra clase *synchronized* de los métodos. El resultado puede sorprender, ya que puede aumentar la velocidad hasta un 40%. En caso de que no se pueda eliminar la sincronización, se puede mejorar el rendimiento situando la palabra clave *synchronized* en bloque, en vez de en la *signature* de método. Por ejemplo, en

```
for ( i=0 ; i < ... )  {
      x.f(); // f() esta sincronizado para x
}
```

se puede hacer:

```
synchronized (x){
for ( i=0 ; i < ... ){
     x.f(); // f() NO esta sincronizado
}
```

Recordad que esta modificación debería aplicarse a bloques pequeños, ya que podría incrementarse el número de bloqueos en aplicaciones *multithread*.

CREACIÓN DE OBJETOS Y EL RECOLECTOR DE BASURA

Gracias al recolector de basura, la mayoría de los desarrolladores se olvidan de la creación de objetos en memoria, y de su posterior eliminación. Pero aunque esto es una gran simplificación de desarrollo, también tiene el inconveniente de degradar el rendimiento. Los programadores tienden a crear demasiados objetos, pues son muy cómodos de utilizar, y no hay que preocuparse mucho de ellos, su periodo de vida lo gestiona la máquina virtual, pero esto baja el rendimiento de la aplicación.

La creación y destrucción de objetos en *Java* es muy costosa en rendimiento. Según el tipo de objeto, si tiene constructor, si no lo tiene, si es una clase base o es una derivada, los tiempos pueden variar sensiblemente. Se puede afirmar que el coste de la creación de un objeto es tres veces mayor que el de eliminar un objeto de un *array, o* de un objeto *Vector* con los métodos sincronizados. Es un precio muy alto.

Es admisible que la creación tome su

```
public void f(int cap)
{
    try
    {
        int i = 0;
//    0    0:iconst_0
//    1    1:istore_2
       do
       {
           p[i] = 0;
//    2    2:aload_0
//    3    3:getfield        #4   <Field int[] p>
//    4    6:iload_2
//    5    7:iconst_0
//    6    8:iastore
            i++;
//    7    9:iinc            2  1
       }
       while(true);
//    8   12:goto            2
    }
    catch(ArrayIndexOutOfBoundsException _ex)
//*   9   15:pop
    {
        return;
//   10   16:return
    }
}
```

Figura 3. - *Bytecode* recorrido bucle.

tiempo, pero también ahorramos con la eliminación, la realiza el recolector de basura, no mi programa. Bien, visto así, correcto, pero el recolector de basura también penaliza la ejecución de nuestra aplicación, al ser un *thread* que, aunque está en mínima prioridad, cuando se ejecuta, paraliza nuestra ejecución, ¿no?

Claro que damos por hecho que el recolector actúa según nuestra necesidad (que no siempre es así), y en determinados periodos (el recolector de basura es un sistema no-determinista). Pero, ¿mi servidor puede permitirse el lujo de reducir su rendimiento cada cierto periodo de tiempo? ¿Qué pasa en aplicaciones en las que no se puede parar en determinados periodos? ¿Cómo se aplica un recolector en tiempo real? Las respuestas a estas preguntas no son sencillas, ya que implican un cambio de filosofía en la forma de recolectar basura.

De todos modos, aquellos que hacen servidores de alto rendimiento y abusan de la creación de objetos, con la consiguiente degradación pasado un tiempo, pueden aplicar estas técnicas:

1. Cuando una objeto se crea repetidas veces, por ejemplo, en un determinado método, lo más sencillo, aun en contra de la OOP, es declarar una variable estática en la que se almacena ese objeto, y utilizarlo sucesivas veces. No sólo estamos ahorrando tiempo en la creación del objeto, también en su destrucción. Los métodos estáticos se ejecutan más eficazmente, y sólo ese cambio mejora el rendimiento.

2. En caso de muchas concreciones de objetos, tenemos que variar la táctica. Para ello, nada mejor que usar un objeto *Vector* para añadir todos los objetos que no se utilizan, como si fuera un *pool*. La idea es crear una especie de repositorio de objetos reutilizables. Cuando una concreción de objeto no se necesita, un método estático lo añade al *pool* y de la misma forma, cuando se necesita una concreción, otro método estático busca si existe y lo reutiliza. Se puede construir un método *reinicializar()* para copiar datos, etcétera, ya que el constructor de los objetos se declarará privado para evitar que puedan ser creados accidentalmente.

Aunque pueda parecer que hay demasiado código de gestión en este proceso, esta técnica es muy utilizada en aplicaciones con gran creación de objetos y es sorprendente la mejora del rendimiento, no sólo en velocidad, si no en la mínima degradación, lo que en conjunto hace que la aplicación sea muy estable, mucho más que si no se usa esta técnica. ¡Pregunten, pregunten a los desarrolladores de servidores *Java* con *CORBA*!

COMPROBACIÓN DE ERRORES

El programador de C++ tiene mucho cuidado de no abusar en el uso de excepciones, ya que es el compilador el que genera el código necesario para su tratamiento, y para sorpresa de muchos toda esta información es gestionada en la pila de ejecución. Los desarrolladores en C++ que utilizaban *Visual C++ 5.0* y *CORBA* (*ORBIX*), sabrán muy bien lo que significa esto, ya que un error en la gestión de la pila con excepciones en el *VC++* evitaba la captura de excepciones *CORBA* cuando éstas se propagaban del servidor al cliente, o provocaban errores bastante originales. Sorprendente pero cierto, aunque en la versión *6.0* se ha corregido el error.

En *C++*, muchas comprobaciones de errores penaliza la ejecución, y a parte de esto, el código generado incluye todo el tratamiento de localización del gestor de excepciones. Afortunadamente, igual que en *Java*, las excepciones son del tipo "sin resurrección", por lo que el código añadido es bastante menor.

Por el contrario, la máquina virtual de *Java* trata las excepciones mediante funciones nativas, lo que hace que todo este proceso sea mucho más rápido y eficaz, a parte de hacerse por la propia máquina virtual y no por nuestro código de programa.

Hay que evitar la excesiva creación de objetos

De todos modos, en la mayoría de los proyectos se detecta un abuso de excepciones, que aunque no penaliza el rendimiento en sí, tras un periodo de mantenimiento se ha insertado código basura para tratar ese exceso de excepciones, que sí merma el rendimiento y el mantenimiento. Esto es porque los desarrolladores no tienen muy claro cuándo un problema se debe tratar como una excepción, o como un valor de retorno tradicional.

Aunque la solución a este problema no está escrita en ningún lado, y cada maestrillo tiene su librillo, siempre es bueno ver qué hacen los demás, y en este caso se va a mostrar un ejemplo del código *Java* de *Sun*.

Por definición, las excepciones se utilizan para situaciones anómalas en la ejecución de nuestro programa,(y éstas se pueden o no gestionar en el mismo). Esta definición es muy bonita y sencilla, pero muy imprecisa. Cada desarrollador la interpreta libremente, por eso, no cabe esperar que en un desarrollo la gestión de excepciones sea uniforme, a no ser que un grupo del mismo decida qué política se debe seguir.

Incluso en la jerarquía de *Java* para el tratamiento de excepciones ya surgen pegas con determinadas excepciones, ya no sólo si son *RuntimeException* o no, sino si como tales deberían de existir.

Tomemos el método *read()* de la clase *InputStrea*: *public abstract int read() throws IOException, y* devuelve el siguiente *byte* leído *o, "-1"* si se alcanza el final de fichero. Bien, en este punto ya surge la duda. ¿No existe una *EOFException*?.

El recolector de basura no tiene un comportamiento sistemático

Sigamos un pocos más. Si se lanza una excepción y no es por el final del fichero, ¿por qué entonces?. Veamos qué dice la definición. Se lanza una excepción del tipo *IOEXception* si ocurre un error. Pues muy claro no es. ¿Qué tipo de error se controla? ¿un hachazo al disco duro? ¿qué se intenta leer cuando se alcanza el final del fichero? ...

Pues efectivamente, aunque no parezca coherente, lo es. Cuando se alcanza el final de fichero no está surgiendo ninguna circunstancia anómala, encontrar el final suele ser bastante normal, si se empieza a leer, alguna vez acabará. Si mediante el retorno de un valor se controla esta circunstancia, ¿para que añadir una excepción?

Pues no se añade una excepción porque el final de fichero no es una anomalía en el programa, pero sí lo es que no se pueda leer un archivo porque en un determinado momento quede bloqueado, o que alguien borre una parte cuando lo estoy leyendo o que éste esté corrupto en el disco y no se pueda leer, o que intente leer cuando ya no existen más datos y se encontró el final del fichero.

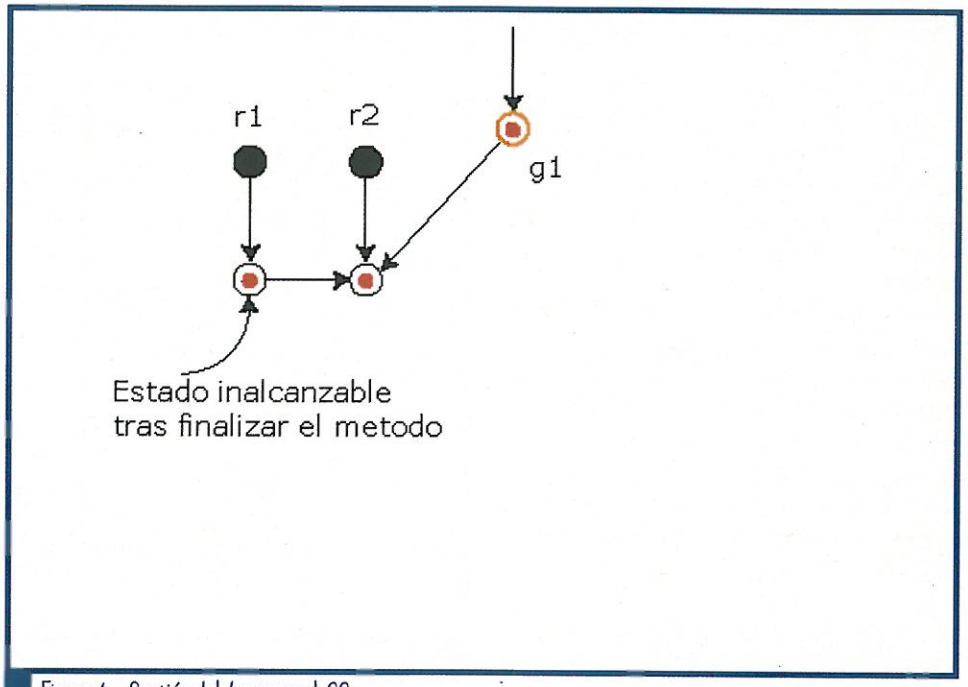


Figura 4. - Gestión del *heap* por el *GC*.

Veamos otro caso algo más aclaratorio: la sentencia *readInt*() en la clase *DataInputStream,* que lee *bytes* y devuelve un tipo entero. El método tiene la siguiente definición:*public final int readInt() throws IOException*. En la ayuda sobre el método nos indican que lee cuatro *bytes* y construye el entero correspondiente, y si se encuentra el final del fichero antes de leer los cuatros *bytes* que faltan, se lanza una *EOFException;* claro, eso sí que es una anomalía que nos impide continuar. Además, se lanza una *IOException* si ocurre cualquier tipo de error de entrada/salida.

La conclusión que el lector debe sacar, es que para el buen rendimiento de las aplicaciones necesitamos una clara definición y buen uso de las excepciones, ya que mal utilizadas pueden generar un código difícil de mantener.

De todas formas, las excepciones en *Java,* al ser implementadas nativamente, se resuelven rápidamente. Es más, no suponen un incremento en el código en sí, ya que no producen códigos de operación, sólo una entrada más en una tabla de excepciones que controla el rango de validación dentro de los métodos, por lo que todo el lanzamiento y recuperación es verdaderamente eficaz.

A veces es conveniente saltarse la regla anterior para mejorar el rendimiento, aunque esto suponga oscurecer el código. Por ejemplo, si se desea inicializar un *array* de enteros con un determinado valor, se suele utilizar el clásico bucle:

```
for(int i = 0 ; i < array.length; i ++)    {
      array[i] = valor ;
}
```

Hay un truco para hacer esto mismo más rápido. Se puede llegar incluso al doble de rápido utilizando... ¡sorpresa!, una excepción, evitando así la comprobación en nuestro programa del rango del *array*:

```
try {
for(int i = 0 ;; i ++){
                  array[i] = valor ;
}
} catch (ArrayIndexOutOfBoundsException e) {
}
```

El código no es precisamente claro y fácil de mantener, pero ofrece mejor rendimiento para *arrays* de gran tamaño. La mejora ronda el 15% del tiempo.

En la figura 3 se puede observar el *bytecode* del bucle optimado.

THREADS

Los *threads* han sido un punto controvertido en el problema de la portabilidad de *Java*, puesto según el sistema

operativo, pueden tener un comportamiento diferente.

Los *threads* suelen sobrecargar la memoria, ya que se implementan usando dos pilas, una *Java* y otra nativa o en estilo "C", lo que hace pensar en la cantidad de memoria utilizada en aplicaciones estilo *multithread*, como servidores de *chat*.

Respecto al tiempo entre cambio de contextos o uso de una implementación nativa, los tiempos no son lo suficientemente importantes para penalizar, aunque hay que tener presente también el tipo de aplicación que se quiera desarrollar... de todos modos, son más preocupantes los *interbloqueos* por mal uso de sincronización.

LIBRERÍAS DE PROPÓSITO GENERAL

Las clases estándar de *Java* permiten hacer muchas tareas, desde un *applet* hasta más complejos sistemas. De todos modos, las librería de propósito general no suelen ofrecer un rendimiento elevado, y el caso de *Sun* en su *JDK,* no es precisamente de lo mejor.

Por tanto, el desarrollador debe tener en cuenta que optimar una porción de código y seguir utilizando librerías no optimadas o de bajo rendimiento, carece de sentido.

Si el rendimiento es un problema, podemos buscar no sólo en el código desarrollado, sino también en librerías de terceros.

MEMORY LEAKS

Uno de lo mayores alicientes de *Java*, a parte de su integración con *Internet*, es el poder olvidarse del desafío que ofrecen los *memory leaks,* gracias al recolector de basura.

Como contrapartida, los desarrolladores ven cómo algunas aplicaciones se degradan en rendimiento y observan la aparición de los síntomas clásicos del efecto *memory leaks*, esto es, crecimiento incontrolado de la memoria utilizada, lentitud general, traducido en un bajo rendimiento que termina con un cuelgue del sistema o con uso imposible de la aplicación. ¿Por qué ocurre esto? ¿qué está pasando? Veamos cómo se gestiona la memoria en *Java* y cómo actúa el recolector de basura.

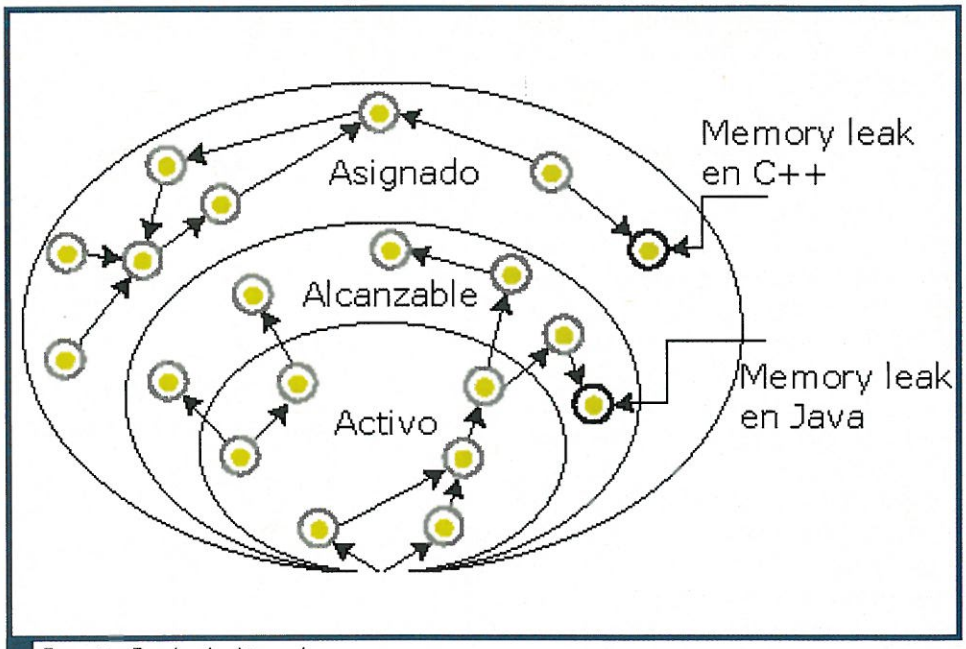


Figura 5. - Estados de objetos *Java*.

Las concreciones de los objetos son asignadas y localizadas en el montículo o *heap* de la aplicación cada vez que se ejecuta una sentencia *new* o se accede mediante referencias.

Se puede describir mediante un grafo dirigido en el que cada nodo lo forma un objeto creado y cada referencia es un conector. El recolector de basura gestiona la memoria para liberar basándose en este tipo de estructura.

Un buen tratamiento de las excepciones mejora el software

Ahora, veamos qué hace el recolector para liberar objetos que supuestamente ya no son necesarios y por tanto disponer de más montículo, sin entrar en cómo se gestiona posteriormente el mismo, que aunque se sale del alcance de este artículo, también es importante.

El problema de determinar cuándo ha dejado de utilizarse un objeto no es trivial, y aunque está más o menos resuelto, su solución es compleja y tiene sus defectos. Como el recolector en un momento determinado no sabe a priori qué objeto se ha dejado de usar (alguien pensará en la indicación de referencia asignada a *null* como pequeña ayuda al recolector) o no se utilizará, se basa en una aproximación, y determina qué objetos no están localizables.

Claro está, la definición de cuándo un objeto es localizable es complicada. Se puede recurrir a la idea del grafo anteriormente presentada. Buscamos los objetos que no puedan ser localizables mediante referencias, a través del grafo de referencias, partiendo de un nodo raíz.

Por tanto, en *Java*, se considera que algo no se utiliza cuando no se puede alcanzar a través de movimientos por el grafo desde cualquiera de los nodos raíz.

Veamos un ejemplo. Supongamos que existen dos referencias locales en un método, *r1* y *r2*, apuntando a sendos objetos. Existe también una variable global *g1* que localiza el objeto que referencia *r2*. El recolector de basura ha determinado que *r1* y *r2* son nodos raíces de su grafo.

Se ejecuta el código y *r1* pasa a apuntar al objeto que apunta *r2*. En este momento, el objeto al que apuntaba *r1* no se puede acceder mediante el grafo, está en un estado digamos de inalcanzable.

Se acaba el método, las referencias se eliminan, y entonces el recolector detecta con el grafo que el objeto al que inicialmente apuntaba *r1* no es accesible, por lo que será eliminado y no podrá ser reutilizado (recuerde la mejora de optimización basada en la reutilización de objetos descrita anteriormente).

Ahora bien, nadie asegura que se elimine en ese momento, puede permanecer ahí durante algún tiempo, incluso hasta que termine el programa si no ha surgido la necesidad de recuperar me-

ıoria. Nunca escriba un programa pre-ıponiendo que el recolector hará el tra-ajo por usted.

Abundan las aseveraciones sobre el :colector, cuya exactitud depende de la nplementación utilizada. Una de ellas s que no trata bien las referencias cícli-ıs, esto es, un objeto *A* apunta a otro *B* éste a otro *C*, y no existen referencias xternas a ellos. No hay ningún tipo de roblema. El recolector eliminará *A*, y omo éste no tiene sentido, irá por *B* y continuación por *C*.

Pero hay una pega -normal en *Micro-ft-*, ya que utiliza una técnica de conta-) de referencias que sí tiene problemas n estas estructuras cíclicas, de ahí to-) ese misterio. Pero aunque *Microsoft*) resuelva esto, no implica que otros bricantes de máquinas virtuales no lo ıyan solucionado.

Otro problema viene de los progra-adores *C++* que se portan a *Java,* y s el método *finalize()*, al que conside-n el destructor del objeto, cuando no s así. Existen diferencias, algunas bas-ntes sutiles, pero sobre todo hay una ıportante.

La más significativa, el método *fina-:e()* no tiene por qué ser llamado, en ntra de los destructores en *C++*. No : puede depender de *finalize()* para ecutar alguna tarea determinada. Otra ıestión para tener en cuenta es que Ja-ı permite la resurrección de objetos acias a *finalize()*.

Pruebe con ello, sólo hay que asignar referencia del objeto que se va a elimi-ır a otro objeto cualquiera que esté aún vo. ¡Premio!, nuestro objeto a punto de orir aparece de nuevo en el grafo, y el colector le deja vivir tranquilo.

TÉCNICAS GENERALES

os objetos en *Java* pueden estar en tres estados:

1. *Asignado*, el objeto creado aún) ha sido eliminado por el recolector.
2. *Localizable*, asignados que pue-:n ser encontrados por el recolector :sde un nodo raíz.
3. En *activo*, localizables que se es-n usando actualmente.

El recolector de basura sigue la pista os objetos asignados pero no localiza-s. Por el contrario, esos objetos serí-los típicos *memory leaks* de *C++*, rque se han perdido hasta que finaliza

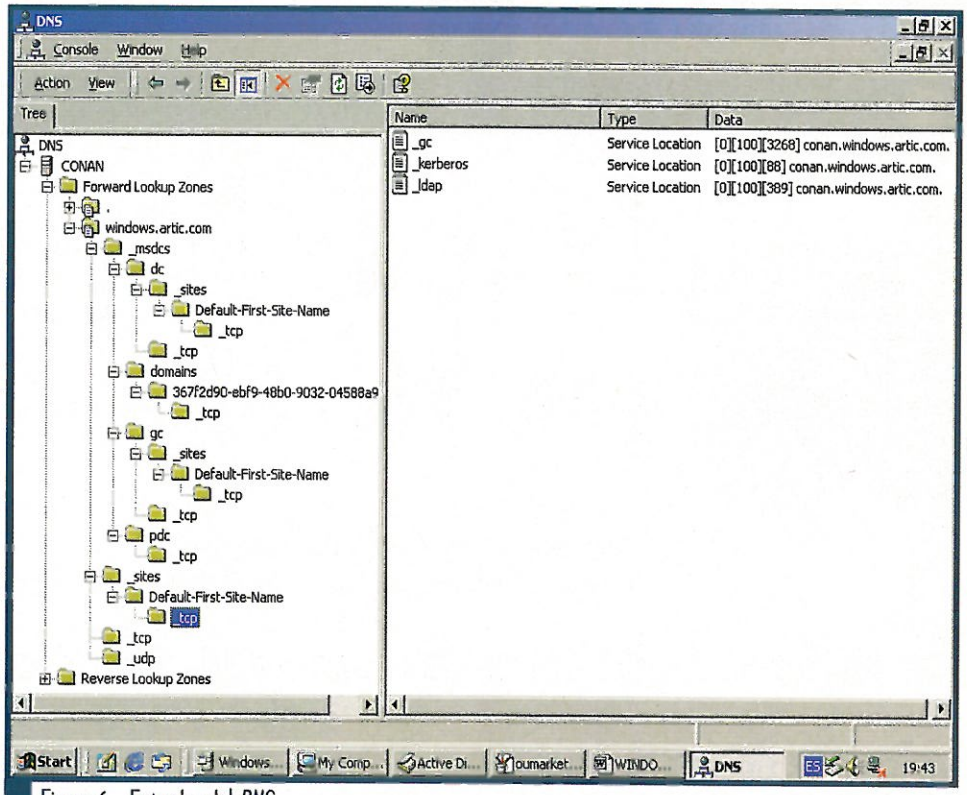


Figura 6. - Entradas del *DNS*.

el programa. En *Java*, un *memory leak* es un objeto cuyo estado es *localizable* pero no está en estado *activo*.

Debido a esta diferencia, los *memory leaks* en *Java* se denominan *loiterers*, o conjunto de objetos que están aún en uso en el programa, y para el recolector de basura han quedado descartados.

A esto también se llama *unintentional object retention,* o *loitering objets*. El lector observará que en los programas de medida de rendimiento y análisis de memoria se utiliza el término *memory leaks* como si fuera *loiterers*.

Existen varios tipos de *loiterers*, siendo los más comunes :

- *Lapsed Listeners.*

Un *lapsed listener* aparece al añadir un objeto a un lista o colección de elementos pero nunca es eliminado. El típico ejemplo está en los *event listeners*, que añaden un nuevo *listener* pero nunca es eliminado cuando ya no es necesario. La mayoría de los desarrolladores olvidan que a parte de un *addXXXListener* existe un *removeXXXListener*.

El crecimiento sin control ni punto de referencia de la lista de *listeners* provoca una caída de rendimiento de la aplicación. Este es el clásico problema al usar *Swing* y *AWT*, sobre todo en grandes aplicaciones con mucha carga gráfica.

- *Lingerers.*

Se produce cuando un objeto es utilizado durante la ejecución de un programa, por ejemplo dentro de un bucle, pero su ejecución continúa al salir de éste. Se puede decir que el objeto inicialmente tenía un periodo de vida transitorio (dentro de un determinado bucle), pero no está claro cuándo y como se ha extinguido. El típico caso es el uso de las funciones de impresión. Se pasa a un servicio de impresión el objeto que se desea imprimir y cuando éste ha sido impreso, es eliminado por el servicio, perdiéndose así cualquier referencia posible desde el código que realizaba la llamada inicial.

- *Laggards*, ocurren cuando un objeto cambia de estado manteniendo las referencias a datos de su estado anterior. Suele pasar al implementar el efecto *transmute*.

Hemos visto que las soluciones para mejorar el rendimiento de una aplicación es muy variado. Para problemas de *loiterers,* es recomendable disponer de algún programa específico como los descritos en el número anterior, pero en muchas ocasiones, bastará con alguna sencilla modificación en el código.

Curso de Programación con WAP (V)

Constantino Sánchez Ballesteros.
Técnico Superior de Desarrollo Aplicaciones Informáticas.

Las funciones en *WMLScript* son parte de la unidad de compilación destinadas a la realización de tareas específicas. Veamos cómo se declaran y utilizan, y cómo construir bucles y condicionales.

DECLARACIÓN DE FUNCIONES

Podemos utilizar la declaración de una función para crear un nombre (identificador) con parámetros opcionales, y el bloque de instrucciones que se ejecutarán dentro de dicha función. Todas las funciones tienen las características:

- Sus nombres son únicos en una unidad de compilación.
- Todos sus parámetros se pasan por valor (no por referencia).
- Las llamadas a funciones deben pasar tantos argumentos a la función llamada como se haya especificado en su declaración.
- Ejecutan el bloque de declaraciones.
- Una función siempre retorna un valor. Por defecto, será una cadena vacía ("").

Podemos utilizar la instrucción *return* para especificar otros valores de retorno en las funciones.

La sintaxis para su declaración es: *f...on nombre_funcion (Lista formal ...ímetros) Block ;opt.*

...emos utilizar la clave opcional ... para especificar que la función será accesible externamente. Las funciones declaradas como externas pueden ser llamadas desde fuera de la unidad de compilación en la que se han definido. *Nombre_funcion* es el nombre utilizado para identificar a la función, a igual que en otros lenguajes de programación. *Block* es el cuerpo de la función que se ejecuta cuando es llamada, y los parámetros han sido inicializados por los

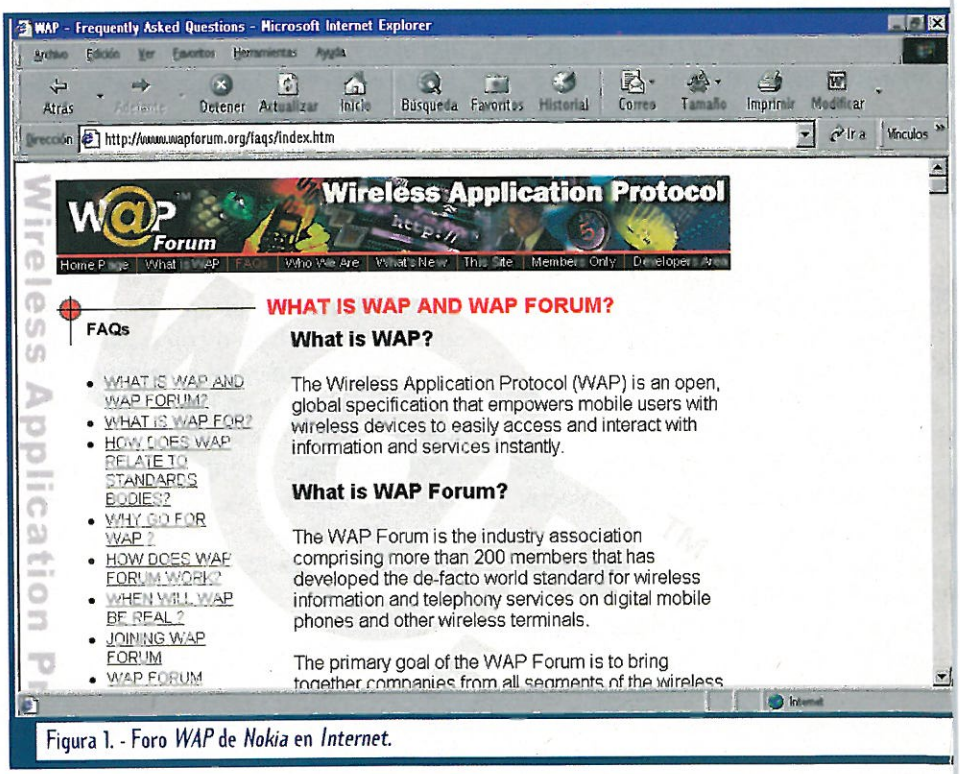


Figura 1. - Foro *WAP* de *Nokia* en *Internet*.

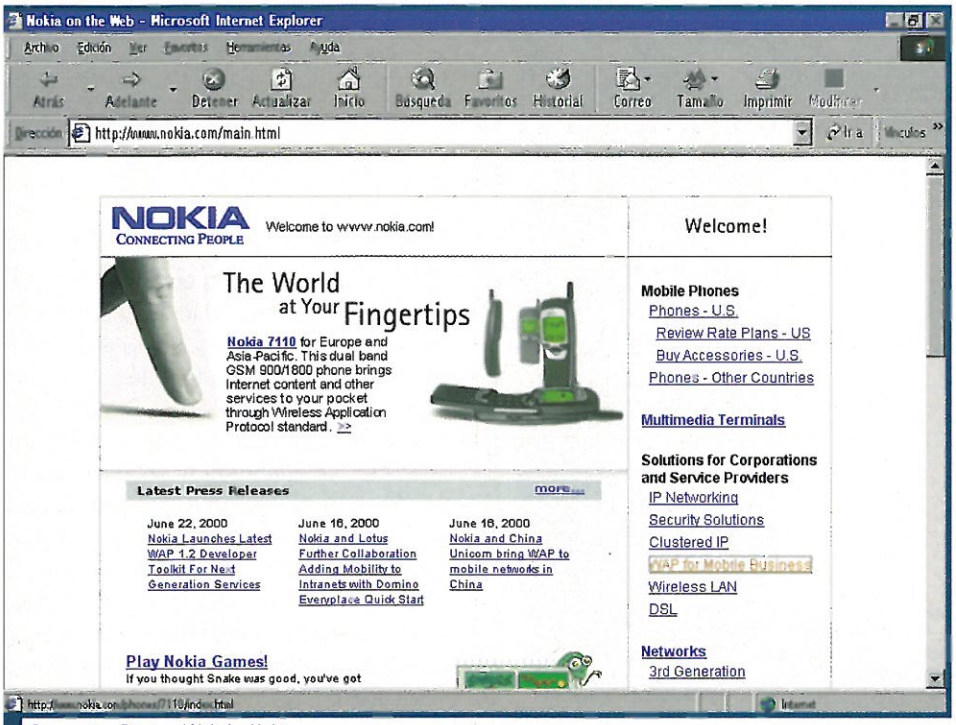


Figura 2. - Página *Web* de *Nokia*.

argumentos que se pasen. En el siguiente ejemplo podemos ver la declaración de una función en *WMLScript*:

```
function calcula_total (pesetas, cantidad) {
return pesetas*cantidad;
};
extern function testear() {
var monedas = 20;
var FIM = calcula_total(monedas, 100);
};
```

En el ejemplo declaramos la función *calcula_total*, que se encarga de multiplicar *pesetas* por *cantidad*, ambos valores pasados como parámetros. Posteriormente, declaramos la función externa *testear*, que internamente realiza una llamada a *Calcula_total* para realizar la operación. *Testear* será la función que utilizaremos desde fuera de la unidad donde se han declarado estas funciones.

LLAMADAS A FUNCIONES

La llamada a una función dependerá de dónde se ha declarado. Las funciones locales se definen dentro de la misma unidad de compilación. Podemos llamarlas escribiendo el nombre de la función y una lista de los argumentos separados por comas. Por supuesto, hay que enviar el mismo número de argumentos que los declarados en la función. Podemos llamar a funciones desde dentro de la misma unidad de compilación aún antes de haber sido declaradas:

```
function test2(parametro) {
return test1(parametro+1);
};
function test1(valor) {
return valor*valor;
};
```

Las llamadas a funciones externas deben hacerse cuando la función a la que se llama se declara en una unidad de compilación externa. Se debe utilizar el *pragma "url"* para especificar la correspondiente unidad de compilación externa:

```
use url mi_Script "http://www.acme.com/script";
function test3(parametro) {
return mi_Script#test2(parametro+1);
};
```

FUNCIONES DE LIBRERÍAS

Las llamadas a funciones de librerías deben usarse cuando la función llamada es una función de la librería estándar de *WMLScript*. Para llamar a una función de la librería estándar debemos utilizar el nombre de la librería seguido por un punto (.) y a continuación el nombre de la función:

```
function test4(parametro) {
return Float.sqrt(Lang.abs(parametro)+1);
};
```

LAS FUNCIONES SON LA BASE PRINCIPAL EN LA CREACIÓN DE SCRIPTS

En este ejemplo, utilizamos la librería *Lang* y la función *abs* para obtener el valor absoluto del parámetro pasado. También utilizamos la librería *Float* y la

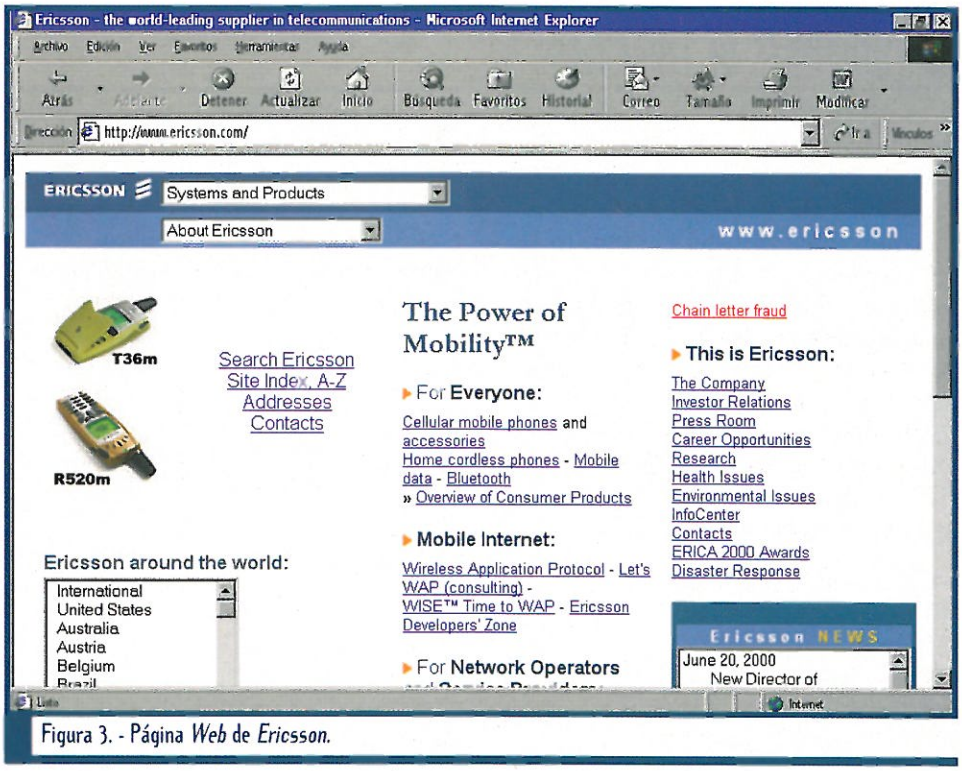


Figura 3. - Página *Web* de *Ericsson*.

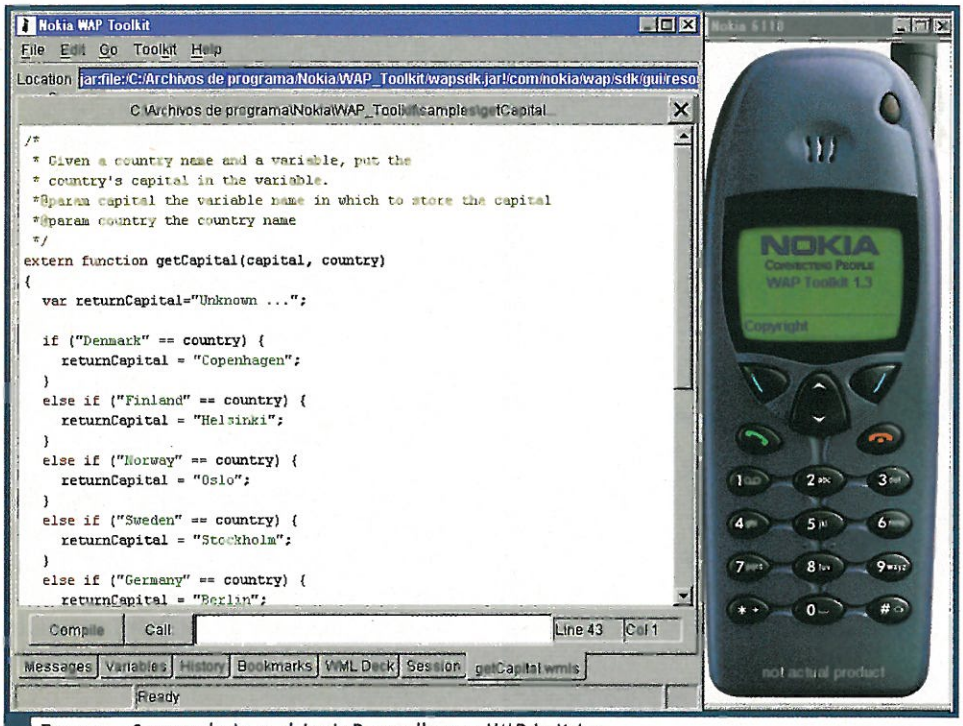


Figura 4. - *Script* incluido en el *kit* de Desarrollo para *WAP* de *Nokia*.

función *sqrt* para calcular la raíz cuadrada. El valor de retorno por defecto para una función, es una cadena de caracteres vacía (""). Estos valores de retorno se pueden ignorar sin ningún problema para la ejecución del código:

```
function test5() {
test4(4);
};
```

FUNCIONES DE LIBRERÍAS

Las declaraciones de *WMLScript* consisten en expresiones y palabras clave con una sintaxis determinada. Una simple declaración puede ocupar múltiples líneas, y múltiples declaraciones pueden entrar en una sola línea. Por ejemplo, la declaración nula se utiliza en los casos donde una declaración se necesita, pero sin efectuar ninguna operación: *while (!test1(valor)) ;* Las expresiones se utilizan para asignar valores a variables, realizar cálculos matemáticos, llamadas a funciones, etcétera:

```
cadena = "Hola " + nombre;
val3 = Val_previo + 4;
contador++;
valor1 = contador, valor2 = val3;
alert("Atención!");
retVal = 16*Lang.max(val3,contador);
```

BLOCK

Una declaración *block* es un conjunto de declaraciones encerradas entre llaves {}, es decir, un bloque de instrucciones:

```
{
var i = 0;
var x = Lang.abs(b);
mensaje("Atención!");
}
```

VAR

Se usa para declarar variables con un inicio. Este inicio es opcional. Las variables se inician por defecto, a una cadena de caracteres vacía (""). Hay que tener en cuenta que los nombres de variables deben ser únicos dentro de una función:

```
function contar(str) {
var result = 0; // se inicializa una vez
 while (str != "") {
  var ind = 0; // Se inicializa multiples veces
   // cada vez que se modifica la string
 };
return result
};

function ejemplo(param) {
var a = 0;
if (param > a) {
var b = a+1; // Variables a y b pueden utilizarse
} else {
var c = a+2; // Variables a, b y c pueden utilizarse
};
return a; // Variables a, b y c son accesibles
};
```

IF

Se utilizan para especificar ejecuciones de código condicionales. Consisten en una condición y una o dos declaraciones, ejecutando la primera declaración si la condición especificada se cumple. En caso de que la condición sea falsa, se ejecutará la segunda declaración, que es opcional: *if (Expresión) declaración else declaración; if (Expresión) declaración.* La *Expresión* (condición) puede ser cualquier expresión *WMLScript* que evalúe directamente, o después de una conversión a *booleano* o valor nulo.

PARA LA CREACIÓN DE BUCLES UTILIZAMOS FOR o WHILE

Si la condición evaluada retorna verdadero, se ejecutará la primera declaración. Si la condición evaluada retorna falso o nulo, se ejecutará la segunda declaración seguida del *else*.

```
if (dia="lunes") {
miDia = "primer dia de la semana";
} else
miDia = "el dia introducido no es el lunes.";
```

WILE

Se utilizan para crear bucles siempre que se dé cierta condición. El bucle dejará de ejecutarse cuando no se cumpla la condición:

```
var contador = 0;
var total = 0;
while (contador < 3) {
 contador++;
 total += contador;
};
```

FOR

Se utilizan para crear bucles de los que se sabe su duración desde un principio, y consta de tres expresiones. La primera se utiliza para iniciar la variable *contador*. La segunda, es la condición que se tiene que dar para que el bucle no acabe. La tercera expresión, es el

incremento, y se encarga de actualizar el contador de la primera expresión:

```
for (var index = 0; index < 100; index++) {
count += index;
miFuncion(count);
};
```

LA LIBRERÍA LANG PERMITE GENERAR NÚMEROS ALEATORIOS

La declaración *break* es utilizada tanto en bucles *while* como en *for*. Se encarga de cerrar el bucle actual y continúa la ejecución donde acaba la declaración del bucle. Hay que prestar atención en no poner la declaración *break* fuera de un bucle *while* o *for*, porque incurriríamos en un error de sintaxis.

```
function testBreak(x) {
var index = 0;
while (index < 6) {
if (index == 3) break;
index++;
};
return index*x;
};
```

CONTINUE

Se encarga de terminar la ejecución de un bloque de declaraciones dentro de un *while* o *for*, y continúa la ejecución del bucle dándole una nueva vuelta e ignorando las instrucciones siguientes a *continue*.

LAS LIBRERÍAS ESTÁNDAR TIENEN FUNCIONES PARA MUCHAS IMPLEMENTACIONES

Hay que tener en cuenta que *continue* no termina la ejecución de un bucle, sino que pasa el bucle de nuevo. Del mismo modo que *break*, *continue* tampoco puede utilizarse fuera de un *while* o *for*.

```
var index = 0;
var count = 0;

while (index < 5) {
index++;
if (index == 3)
 continue;
 count += index;
};
```

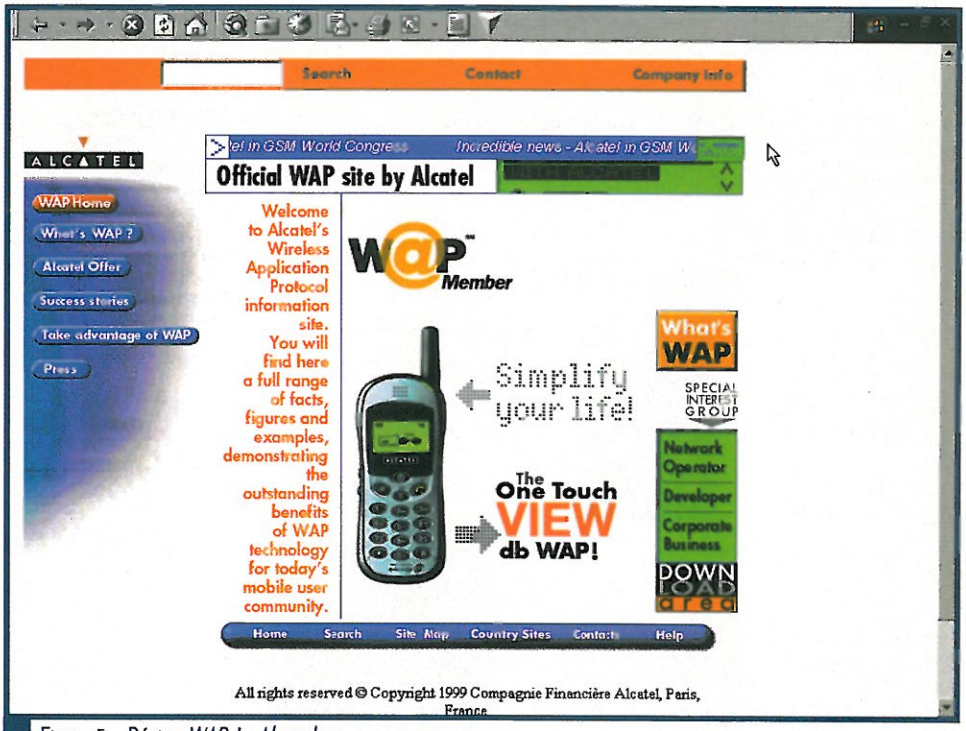


Figura 5. - Página *WAP* de *Alcatel*.

RETURN

Podemos utilizarla dentro del cuerpo de la función para especificar el valor de retorno de dicha función. Si no se especifica *return*, la función retornará por defecto una cadena de caracteres vacía.

```
function cuadrado( x ) {
if (!(Lang.isFloat(x))) return invalid;
return x * x;
};
```

LIBRERÍAS

WMLScript como, ya se ha dicho, soporta el uso de librerías. Éstas, son colecciones de funciones. Podemos hacer una llamada a cualquiera de estas funciones incorporadas utilizando el nombre de la librería más el punto y el nombre de la función, con los correspondientes parámetros entre paréntesis. A continuación podemos ver una llamada a una de estas funciones:

```
function prueba(str) {
var i = String.elementAt(str,3," ");
};
```

En este caso, se ha utilizado la librería *string* para el manejo de cadenas de caracteres, y la función *elementAt*, la cual necesita tres parámetros para funcionar correctamente.

LIBRERÍAS ESTÁNDAR

En este apartado vamos a ver algunas funciones incluidas en librerías estándar de *WMLScript*.

LANG

Esta librería contiene funciones que se utilizan comúnmente en *WMLScript*. Podemos generar números aleatorios, obtener valores absolutos de números, etcétera.

ABS

Función: abs(*valor*)Descripción: Retorna el valor absoluto de un número dado. Si el número es un entero, la función retornará un número entero. Del mismo modo, si el número es en coma flotante, se retornará un número en coma flotante. Parámetros: *valor*=Número. Valor de retorno: número o nulo. (*var a = -1; var b = Lang.abs(a);*)

El resultado de este ejemplo será que *b* contendrá el valor de *1* en lugar del *–1* enviado para procesar en la variable *a*.

MIN

Función: min(*valor1, valor2*). Descripción: Retorna el valor mínimo de los dos números pasados como parámetros. El valor y el tipo retornado son iguales

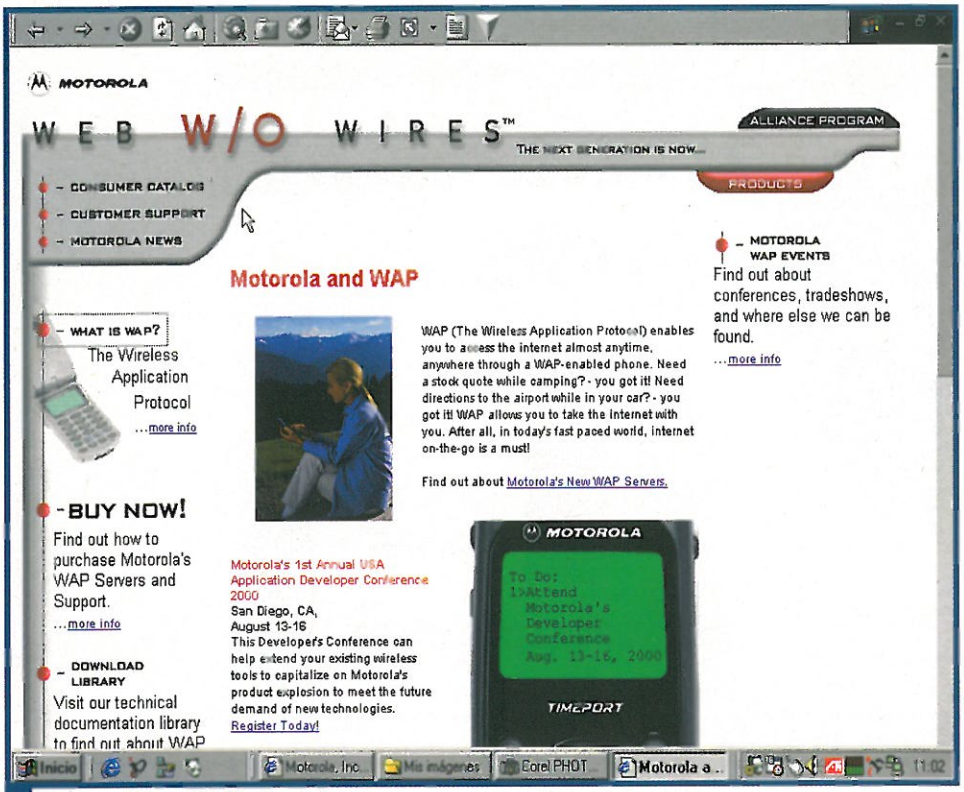


Figura 6. - Página *WAP* de Motorola.

que el valor y tipo del número seleccionado. Parámetros:

*valor*1=Número; *valor*2=Número; valor de retorno: Número o nulo(*var a=-3; var b=Lang.abs(a); var c=Lang.min(a,b); // c=-3*).

En este extracto de código, la variable *b* obtendrá el valor absoluto de *a*, que será *3*. Al utilizar *min*, *c* obtendrá el valor menor entre *a* y *b*, el cual será **–3**:

```
var d=Lang.min(45, 76.3); // d=45 (entero)
var e=Lang.min(45, 45.0); // e=45 (entero)
```

MAX

Función: max(*valor1, valor2*). Descripción: al contrario que *min*, retorna el valor máximo entre dos números dados:

```
var a=-3;
var b=Lang.abs(a);
var c=Lang.max(a,b); // c=3
var d=Lang.max(45.5, 76); // d=76 (entero)
var e=Lang.max(45.0, 45); // e=45.0 (decimal)
```

PARSENT

Función: *parseInt*(*valor*). Descripción: retorna un valor entero convirtiendo la cadena de caracteres. La conversión acaba cuando el primer carácter que se encuentra no es el signo (+) o (–), sin paréntesis, o un dígito decimal.

RANDOM

Función: *random*(*valor*).Descripción: retorna un valor entero positivo mayor o igual a cero pero menor o igual que el valor dado. El valor retornado es elegido de forma aleatoria o pseudoaleatoria. Si el valor es decimal, primero se utilizará la función *Float.int*( ) para calcular el valor entero del decimal pasado. Parámetros: *valor*=Número entero. Valor de retorno: entero o nulo:

```
var a=10;
var b=Lang.random(5.1)*a; // b=0..50
var c=Lang.random("string"); // c=nulo
```

ABS

Función: *seed*(*valor*). Descripción: Inicia la secuencia de números aleatorios (la semilla) y retorna una cadena de caracteres vacía.

Si el valor dado es cero o un número entero positivo, *valor* se utilizará para el inicio. Parámetros: *valor*=Número entero. Valor de retorno: *String* o nulo:

```
var a=Lang.seed(123); // a=""
var b=Lang.random(20); // b=0..20
var c=Lang.seed("seed"); // c=NULO
```

En la próxima entrega, última de la serie, veremos algunas librerías estándar adicionales y crearemos varios programas que utilicen funciones dentro de *scripts* creados en *WMLScript*.

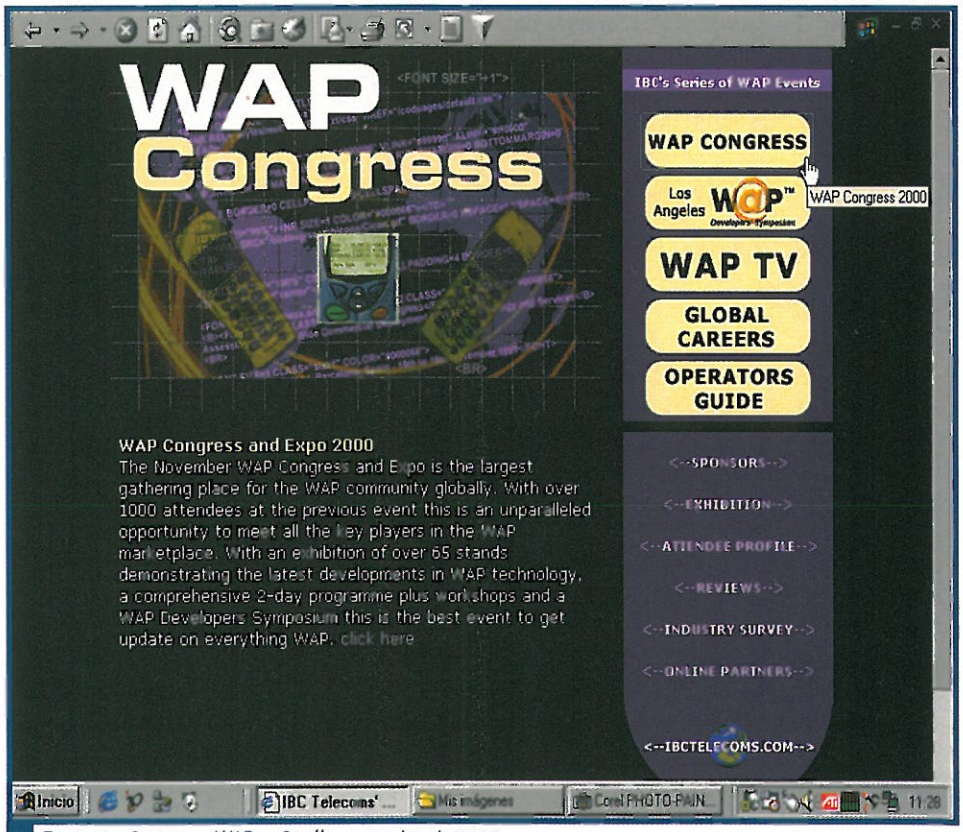


Figura 7. - Congreso *WAP* en Sevilla, noviembre de 2000.

DE RIENDA SUELTA A SU CREATIVIDAD E IMAGINACIÓN CON:

MASTER COREL DRAW

HASTA UN 55% DE DESCUENTO

Suscríbase ahora al Master Corel Draw, y junto a la colección de 50 fascículos y 50 CD-ROM recibirá la *Suite completa de Corel Draw 9.0, versión estudiante.*

Más de 1.000 páginas de teoría y práctica llenas de los más útiles consejos para convertirse en un auténtico Master del diseño gráfico.

Más de 200 programas y utilidades del diseño gráfico, más de 500 fotografías profesionales y decenas de Webs de diseño para navegación off-line.

Ejercicios basados en las explicaciones del curso que le ayudarán a perfeccionar su técnica.

Diploma acreditativo de haber recibido y aprovechado el *Master Corel Draw* de diseño gráfico.

Con 5 carpetas para almacenar sus fascículos y un archivador para ordenar los CD-ROM.

Desarrolle toda la capacidad de las nuevas herramientas que le ofrece Corel Draw 9: Ilustraciones vectoriales, Corel Photo-Paint 9 (edición de imágenes y bitmaps), C. Texture (Texturas realistas y naturales), C. Capture (Aplicación para la captura de pantallas), 25.000 imágenes clip-art de alta calidad (10.000 nuevas para C. Draw 9), y muchas otros extras.

RECORTE Y ENVÍE EL CUPÓN ADJUNTO A:

Revistas Profesionales, S. L. • San Sotero, 5-1.ª planta • 28037 • Madrid • Fax: 91 327 13 03

BOLETÍN DE SUSCRIPCIÓN — OFERTA VÁLIDA SÓLO PARA ESPAÑA

SI, deseo suscribirme al **MASTER COREL DRAW**

CONDICIONES DE PAGO (MARQUE LA OPCIÓN DESEADA)

50 fascículos + 46 CD-ROM 795 Ptas/u. **TOTAL: 39.750 PTAS.**
5 carpetas para archivar fascículos a 950 Ptas./u. **TOTAL: 4.750 PTAS.**
1 carpeta para archivar CD's a 1.250 Ptas.

VALORACIÓN TOTAL DE LA COLECCIÓN CON COREL DRAW 9 INCLUIDO: 84.750 PTAS.

Pago al contado. Ahorro del **55%**. **TOTAL: 38.140 Ptas.**
Pago en 3 plazos. Ahorro del **45%**. Cada plazo: 15.535 Ptas. TOTAL: 46.610

ENTREGA DEL COREL DRAW 9 EN EL ÚLTIMO ENVÍO

n Corel Draw 9 valoración total: 45.750 Ptas.

Pago en 6 plazos (con ahorro del **25%**). Cada plazo: 5.720 Ptas. TOTAL: 34.315
Pago en 12 plazos (con ahorro del **20%**). Cada plazo: 3.050 Ptas. TOTAL: 36.600

TILICE MAYÚSCULAS PARA RELLENAR ESTA TARJETA

ombre y apellidos ..... F. nacimiento.....
omicilio ..... C.P. .....
iudad ..... Provincia ..... Telf. .....
-mail ..... Profesión .....

FORMAS DE PAGO:

❑ Con **cargo a mi tarjeta VISA** Nº ____/____/____/____ Caduca .....

❑ **Domiciliación bancaria**

Ruego a Ud. que se sirva cargar en mi ❑ cuenta corriente ❑ libreta de ahorro el/los recibos que le serán presentados por REVISTAS PROFESIONALES, S.L. como pago de mi suscripción al **MASTER COREL DRAW DE DISEÑO GRÁFICO**

CÓDIGO CUENTA CORRIENTE

ENTIDAD	OFICINA	DC	Nº CUENTA

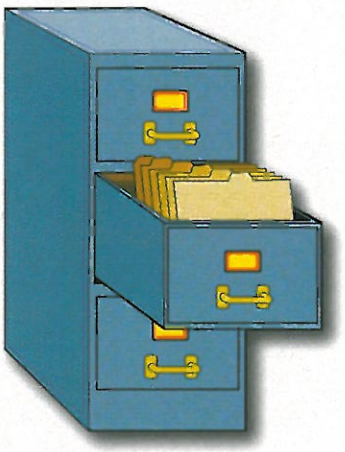
❑ **Cheque** a nombre de REVISTAS PROFESIONALES, SL.

❑ **Contra reembolso** del importe más gastos de envío.

❑ **Giro Postal** (adjunto fotocopia del resguardo)*.

Firma

*Esta forma de pago es exclusivamente para pagos al contado



Aplicaciones de bases de datos con Delphi 5 (y VII)

Juan Luis Ceada Ramos.
Ingeniero Superior de Informática.

Terminamos la serie sobre bases de datos en *Delphi 5*, viendo cómo utilizar las actualizaciones en *caché* para mejorar el rendimiento.

COMPONENTES IBX

Antes de continuar con las explicaciones, conviene saber que existe un parche para los componentes *IBX*, que se puede descargar desde www.interbase.com o www.interbase2000.org:

TIBTABLE y TIBQUERY: Estos componentes son casi idénticos a sus equivalentes *BDE* (*TTable* y *TQuery*), por lo que remito al lector a anteriores entregas. La única diferencia importante es que ahora estos componentes tienen una propiedad *Transaction*, a la que hay que asignar el nombre de un componente *TIBTransaction* (sólo en caso de que usemos las transacciones, claro está).

TIBStoredProc y TIBUpdateSQL: El primero es equivalente al componente *TStoredProc*, excepto por la ya mencionada propiedad *Transaction*. El segundo, a su vez, es idéntico al componente *TUpdateSql* (del cual hablaremos un poco más adelante).

TIBDataSet: Este componente no tiene equivalente en el *BDE*, aunque sí en los componentes *ADO*. Básicamente, es idéntico a *TIBQuery*, aunque este componente permite trabajar con conjuntos de datos provenientes de una consulta *SQL* no actualizable, permitiendo la edición de los mismos. Es como si combinásemos un *TIBQuery* con un *TIBUpdateSql* en un único componente.

Una consulta es sólo de lectura cuando presenta datos de varias tablas simultáneamente. Gracias a las propiedades *DeleteSQL*, *ModifySQL* e *InsertSQL*, podremos trabajar con estos datos como provenientes de una única tabla. *Delphi* recurrirá a estas propiedades cuando

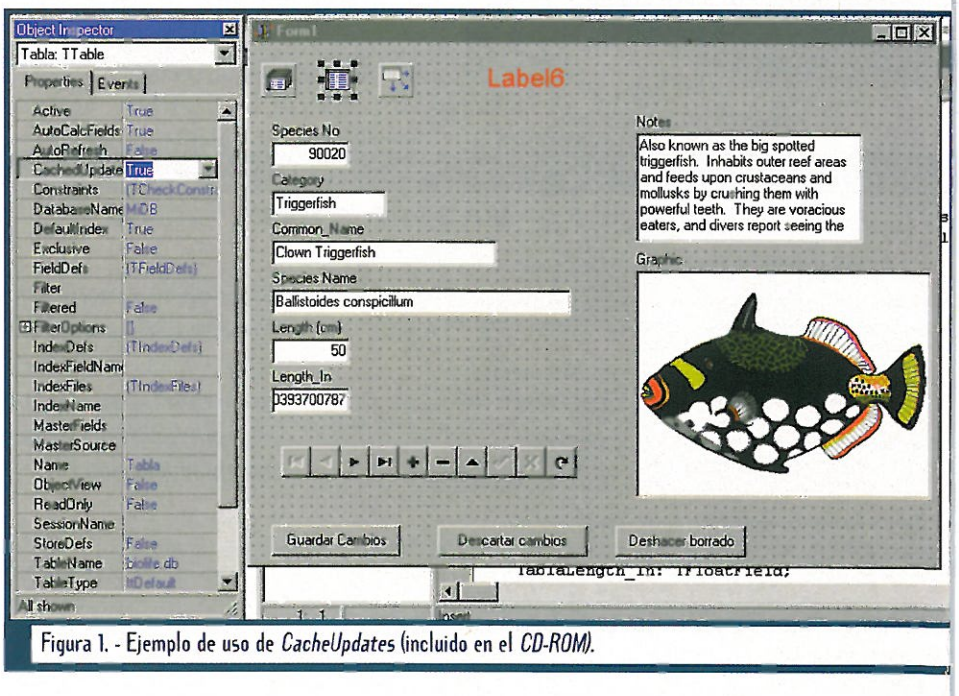


Figura 1. - Ejemplo de uso de *CacheUpdates* (incluido en el *CD-ROM*).

tenga que insertar, añadir o modificar algún registro.

TIBSQL: Permite ejecutar sentencias *SQL* que no devuelvan resultados (por lo que no se puede enganchar a un *TDataSource*), como por ejemplo *insert's*, *delete's*, sentencias de creación de tablas, etcétera.

La diferencia entre usar este componente y un *TIBQuery* para ejecutar este tipo de sentencias *SQL* reside sobre todo en la velocidad de ejecución: sencillamente, *TIBSql* es muchísimo más rápido. Por lo demás, se usa igual que *TIBQuery* (y por tanto, igual que *TQuery*).

Los IBX son muy parecidos a sus equivalentes BDE

TIBDatabaseInfo: Permite obtener información de la base de datos a la que esté conectado (indicada por la propiedad *Database*): características de la base de datos y del entorno, estadísticas de rendimiento y valor de los contadores internos. Para usarlo, basta con insertarlo en el formulario, no siendo necesario llamar a ningún método. Para obtener la información, tendremos que recurrir a algunas de sus múltiples propiedades. Remito al lector a la ayuda de *Delphi* para obtener un listado de todas las propiedades disponibles. Como ejemplo, citaremos la propiedad *CurrentMemory,* que devuelve la cantidad de memoria que está usando el servidor, y *UserNames,* que devuelve el nombre de todos los usuarios conectados actualmente a la base de datos.

TIBSQLMONITOR: Este componente permite controlar el tráfico que nuestra aplicación genera hacia el gestor de base de datos. Cuando la propiedad *TraceFlags* está en *true,* se controla toda la actividad de la conexión con la *BD,* que puede ser enviada a un fichero, o a algún componente de tipo *TList* (un *TListBox,* por ejemplo). Si quisiésemos ver todos los mensajes en un *TListBox,* bastaría con añadir *ListBox1.Items.Add (EventText);* en el evento *OnSQL* del componente.

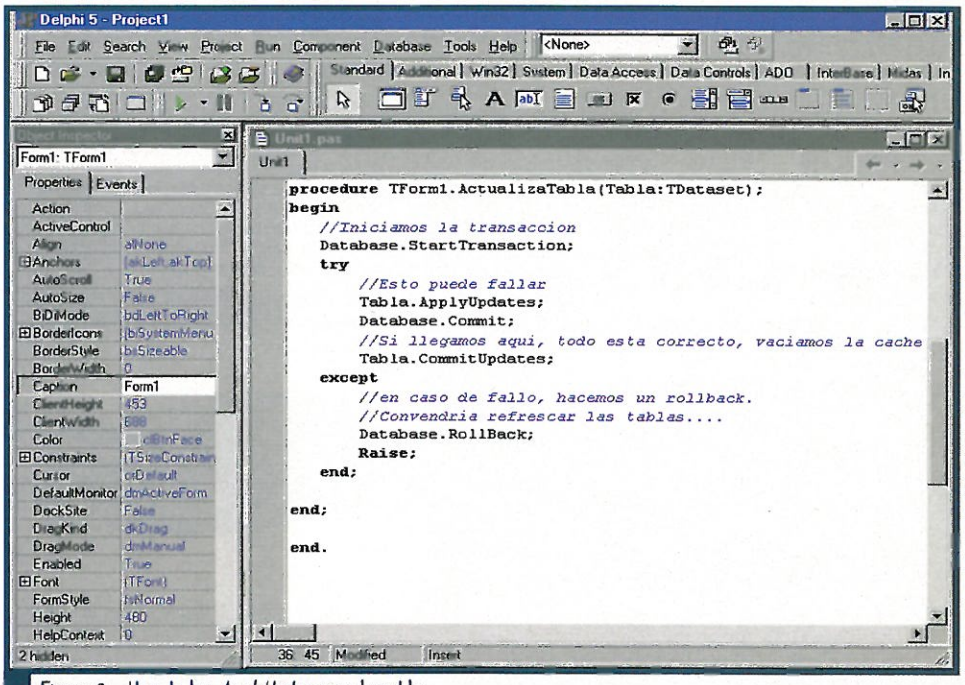


Figura 2. - Uso de los *ApplyUpdates* en la tabla.

OTROS COMPONENTES: Desde *www.interbase.com* es posible descargar una nueva versión de los componentes *IBX* (compatibles con *Interbase* 6) que añade una pestaña de componentes al *IDE* de *Delphi,* llamada Interbase *Admin Tab,* en la que se localizan una serie de componentes para mantener y configurar las *BBDD.*

Podéis encontrar una descripción resumida de cada uno de ellos en la Tabla 1.

TABLA 1. Nuevos componentes IBX para Delphi 5 e Interbase 6.

Componente	Descripción
TIBConfigService	Modificar *buffers,* acceso a *BDD* etc.
TIBBackupService	Permite realizar *backups* de las *BBDD.*
TIBRestoreService	Permite restaurar copias de seguridad.
TIBValidationService	Validar la *BDD* y recuperar transacciones "huérfanas".
TIBStaticalService	Consultar estadísticas
TIBLogService	Permite generar ficheros de *logs.*
TIBSecurityService	Control de acceso a usuarios.
TIBLicensingService	Edición de licencias instaladas.
TIBServerProperties	Propiedades de configuración del servidor, versión, e información de las licencias instaladas.
TIBInstall	Permite crear instalaciones.
TIBUnInstall	Desinstalar instalaciones previas

También podéis consultar la documentación del nuevo *Interbase* 6.0 (que por cierto, es *Open Source),* en concreto, la *Developer Guide,* en la dirección *web* indicada.

ACTUALIZACIONES EN CACHÉ

Supongamos que hemos desarrollado una aplicación cliente/servidor con la que van a trabajar varios usuarios simultáneamente. Es muy posible que dichos usuarios trabajen con los mismos registros a la vez. Los sistemas gestores de bases de datos tienen que garantizar la coherencia de la información que proporcionan a los usuarios.

Una de las tareas principales de los *SGBD* es garantizar que accesos concurrentes a los datos no provocan inconsistencias en la información. Mediante las transacciones garantizamos que los datos están protegidos ante situaciones catastróficas (por ejemplo, un corte de luz durante una actualización). Pero además, durante las transacciones los *SGBD's* introducen bloqueos sobre los registros (o incluso sobre tablas completas), que permiten garantizar la coherencia de los datos cuando hay accesos concurrentes.

Cada vez que se actualiza un registro, éste se bloquea. Así conseguimos que el resto de usuarios no puedan acceder al

Figura 3. - Registro que aparece como borrado.

mismo, ya puesto que la información almacenada puede ser incorrecta. ¿Cuándo se liberan los bloqueos? Pues cuando confirmamos (o anulamos) la transacción.

Como normal general, cuanto menos dure una transacción, mejor, porque así los bloqueos sobre la base de datos también estarán activos durante menos tiempo, favoreciendo así el uso concurrente de los datos. Las actualizaciones en caché permiten reducir drásticamente la duración de las transacciones, y con ello , la de los bloqueos, aumentando el rendimiento del sistema y reduciendo el tráfico de la red.

¿CÓMO FUNCIONAN?

Pongamos un ejemplo extremo: supongamos que un usuario está actualizando los datos. Con el proceso, comienza una transacción, y el sistema bloquea cientos de registros. Pero de repente, llega la hora del desayuno, y nuestro usuario se marcha sin confirmar la transacción. Durante unos 20 minutos, otras personas no pueden trabajar con esos datos, al estar bloqueados.

Con actualizaciones en caché, los datos viajarían a la máquina del usuario. Este trabajaría con ellos todo el tiempo que quisiese (desayuno incluido). Cuando pulse el botón "Guardar Cambios", empezará la transacción (las actualizaciones en caché se apoyan internamente en transacciones), se actualizaran los datos, y se hará el *commit*. Todo lleva pocos segundos.

Al contrario que en el primer caso, otros usuarios han podido trabajar con los datos mientras nuestro usuario desayunaba. Aunque si otros modificaron los datos en este lapso, se encontrará con que la actualización falla, y seguramente deba repetir el proceso. Para actualizar en caché, basta con poner la propiedad *CacheUpdates* en *true* (presente en *TTables, TQuerys* y similares). Existen además, métodos y propiedades relacionados con las actualizaciones en caché. Por ejemplo, la propiedad *UpdatesPending* indica si existen actualizaciones en caché pendientes de confirmación.

Para confirmar las actualizaciones, contamos con *ApplyUpdates*. Existen dos versiones: *ApplyUpdates,* para base de datos (método de *TDatabase)* y para tabla (método de *TTable).* En el primer caso, es posible pasar un conjunto de *TDatasets* para que sus datos sean confirmados. Por ejemplo, *Database.ApplyUpdates([tblEspecies, tblEspecimenes]),* confirmaría los cambios realizados en las tablas de especies y especímenes. En caso de que alguna de estas tablas esté en modo de edición, la llamada a *ApplyUpdates* realizará un *Post* del registro en edición. Además, internamente inicia y acaba una transacción.

Si se desea usar *ApplyUpdates* en tabla, hay que tener en cuenta que realmente la confirmación de los datos se hace en dos pasos: se graba el contenido de la caché en la base de datos y se confirma la transacción (mediante *Table.ApplyUpdates).* Suponiendo que no haya ningún fallo (esta fase puede fallar por múltiples motivos, como registros duplicados, incumplimiento de la integridad referencial, etcétera), se pasa a la segunda fase. En esta, se vacía la caché, que queda lista para comenzar una nueva operación (mediante *Table.CommitUpdates).* Por supuesto, estas operaciones deben estar dentro de una transacción, si queremos que el comportamiento sea idéntico al del método *ApplyUpdates* del *Tdatabase.* Consulte el código de la Figura 5.

Los IBX para Interbase 6 incluyen nuevos componentes

Para descartar todos los cambios realizados en la caché contamos con el método *CancelUpdates* (esta vez en *TDataset).* De forma análoga a *ApplyUpdates,* este método realiza un *Cancel* del registro actual en caso de que éste se encuentre en modo de edición. Si deseamos cancelar los cambios en todos los *datasets*, podemos usar la siguiente rutina:

```
Procedure Cancelar(DB:TDatabase);
var i:integer;
begin
 For i:=0 to DB.DataSetCount - 1
 do begin
    DB.DataSets[i].CancelUpdates;
 end;
End;
```

OTRAS PROPIEDADES Y MÉTODOS

Mediante la propiedad *UpdateStatus* (de los *TDatasets)* podemos saber cuál ha sido la última actualización hecha en el registro. Los posibles valores son *usUnmodified* (no se ha cambiado), *usModified* (se ha modificado), *usInserted* (es un registro nuevo) y *usDeleted* (el registro se ha borrado). Un registro en cualquiera de estos estados puede volver al estado inicial mediante el método *RevertRecord* (de nuevo perteneciente a los *TDataSets).* Así, podemos recuperar los registros que hayamos borrado durante la operación actual.

Además, si queremos, podemos hacer que los registros borrados sigan a la vista. Podemos hacer que cuando este-

LISTADO1. Código del evento OnUpdateRecord.

```
procedure TForm1.Query1UpdateRecord(DataSet: TDataSet;
  UpdateKind: TUpdateKind; var UpdateAction: TUpdateAction);
begin
  //primero actualizamos los pedidos
  UpdateSQLPedidos.SetParams(
  UpdateKind);
  UpdateSQLPedidos.ExecSQL(
  UpdateKind);

  //y ahora los clientes
  UpdateSQLClientes.SetParams(
  UpdateKind);
  UpdateSQLClientes.ExecSQL(
  UpdateKind);

  //y por ultimo
  UpdateAction:=uaApplied;

end;
```

ıos sobre un registro borrado, se active n botón que nos permita recuperarlo.

Para que los registros borrados sigan la vista del usuario, tenemos que mo- ificar la propiedad *UpdateRecordTypes* el *TDataset*. Es un conjunto que puede bergar las siguientes constantes: *rtMo ified* (mostrar los registros modifica- os), *rtInserted* (mostrar los nuevos re- istros), *rtDeleted* (mostrar los registros liminados), y *rtUnModified* (mostrar los egistros no modificados). Por defecto, e muestran todos menos los borrados, sí que para conseguir que éstos tam- ién se vean, deberemos hacer (por jemplo, en el *OnCreate* del formulario): *abla. UpdateRecordTypes := Tabla.Up- ateRecordTypes + [rtDeleted]*

CONTROL DE ERRORES

Ahora los errores no se controlan mediante *OnPostError* y *OnDeleteError*, no que contamos con un evento especí- co para estos casos: *OnUpdateError*. El rmato de este evento es el siguiente:

```
rocedure Table1UpdateRecord(DataSet:TDataset; E:
    EDatabaseError; UpdateKind: TUpdateKind;
    var UpdateAction:TUpdateAction)
```

Como se ve, se nos indica el *dataset*, excepción que va a producirse y el ti- o de actualización que se estaba reali- ando (una modificación, borrado o in- erción). El parámetro *UpdateAction* es e entrada/salida, lo que nos permite ecidir qué hacer en caso de error.

Los posibles valores que puede to- ar son: *uaAbort* (aborta las actualizaciones), *uaFail* (aborta y lanza una excepción), *uaRetry* (repetir la operación), *uaSkip* (ignorar el registro actual y continuar con el siguiente).

En caso de fallo, deberíamos comprobar de qué tipo de error se trata (consultado la excepción producida). En caso de que podamos eliminarlo, asignaremos a *UpdateAction* los valores *uaRetry* o *uaSkip*, según nos convenga. En caso de que no podamos eliminar el error, lo más lógico es abortar la operación. Un error común puede ser la inserción de registros duplicados (violación de claves primarias), o conflictos debidos a bloqueos (otro usuario modificó los datos mientras nosotros lo hacíamos). En la aplicación de ejemplo se muestra cómo controlar los errores debidos a registros duplicados.

TUPDATESQL

Hay un caso en el que estamos obligados a utilizar las actualizaciones en caché: cuando deseamos que consultas *SQL* que implican a una o más tablas sean editables. En efecto, tal como define el estándar *SQL*, una consulta no es editable si contiene joins (dos o más tablas enlazadas), cláusulas *distinct, group by,* etcétera.

Las actualizaciones en caché mejoran el rendimiento

En estos casos, nuestra única solución consiste en usar un *TUpdateSQL*. Este componente se asocia a un *Dataset* mediante la propiedad *UpdateObject*. Proporciona tres propiedades que permiten introducir las cadenas *SQL* necesarias para modificar, insertar o eliminar registros.

Por ejemplo, supongamos la consulta *Select * from clientes C, pedidos P where C.id_cliente = P.id_cliente*. Esta consulta (que no es editable) se introduce en un *TQuery*. El conjunto de registros devueltos se visualiza en un *TDBGrid*.

Ahora introducimos un componente *TUpdateSQL*, que enlazamos en tiempo de diseño con el *TQuery* (*TQuery.UpdateObject = TUpdateSQL*). Un doble clic sobre el *TUpdateSql* abre el generador de consultas *SQL*. Seleccionamos la tabla que queremos que el componente actualice (la de clientes o la de pedidos), y pulsamos el botón *Generate Sql*. Esto nos creará el esqueleto básico de las consul-

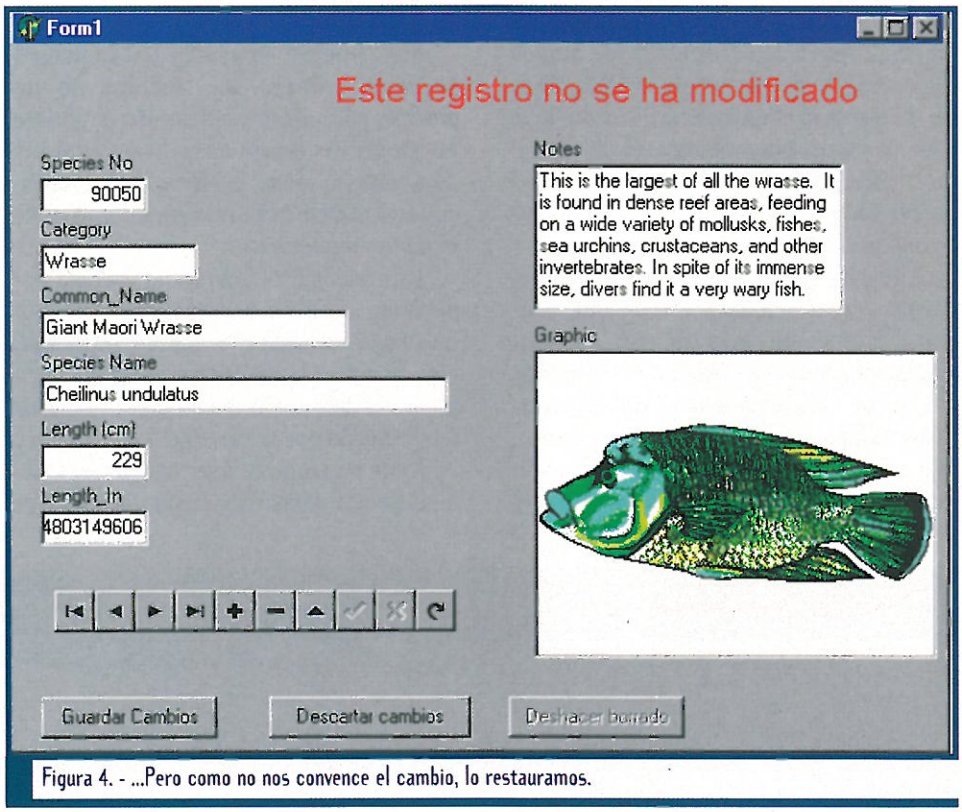


Figura 4. - ...Pero como no nos convence el cambio, lo restauramos.

tas *SQL* necesarias para actualizar, insertar o borrar registros (sólo es un diseño, puede ser necesario modificarlas).

De esta forma, si borramos un registro del *grid* (operación *Query1.delete),* automáticamente *Delphi* ejecutará la consulta *DeleteSQL,* de forma que se borrará el registro de la tabla clientes. De igual forma, si insertamos un nuevo registro *(insert),* se ejecutará la sentencia *InsertSQL,* lo que nos introducirá el registro en la tabla clientes.

Puesto que cada objeto *UpdateSQL* permite actualizar una tabla, lo normal es que necesitemos más de uno. En ese caso, no podemos usar las propiedades *UpdateObject* de los *TDatasets* (y descendientes). Tendremos que asignar, en tiempo de ejecución, a cada *TUpdateSQL* el *TDataset* sobre el que va a trabajar.

Por ejemplo, para actualizar también la tabla *pedidos,* tendríamos que introducir un nuevo *TUpdateSQL,* que enlazaríamos (esta vez en tiempo de ejecución), con el *TQuery,* de la siguiente forma:

```
UpdateSQLClientes.Dataset:=Query1;
UpdateSQLPedidos.Dataset:=Query1;
```

Una vez insertadas (en tiempo de diseño o ejecución) las consultas *SQL* de ambos componentes *TUpdateSQL,* sólo queda ejecutar la aplicación. Ahora, cuando borremos un registro del *grid* (con *query1.delete),* automáticamente se lanzarán las instrucciones almacenadas en las propiedades *DeleteSQL* de ambos *TupdateSQL.* En el caso de la tabla *clientes,* se borra el cliente seleccionado de la base de datos. Sin embargo, el *Updatesql* de la tabla *pedidos* podría perfectamente borrar todos los pedidos que estén asociados al cliente.

Aún nos queda un pequeño detalle. Cuando en la actualización de una consulta interviene más de un *TUpdateSQL,* nos veremos obligados a recurrir al evento *OnUpdateRecord* del *TQuery.* Si introducimos código en este evento, queda en nuestras manos la activación de los componentes. Los pasos que hay que dar cuando este evento salte son los siguientes: pasar los parámetros necesarios a las consultas de los *TUpdateSQL,* ejecutar la consulta, y asignar a la variable *UpdateAction* la constante *uaApplied.*

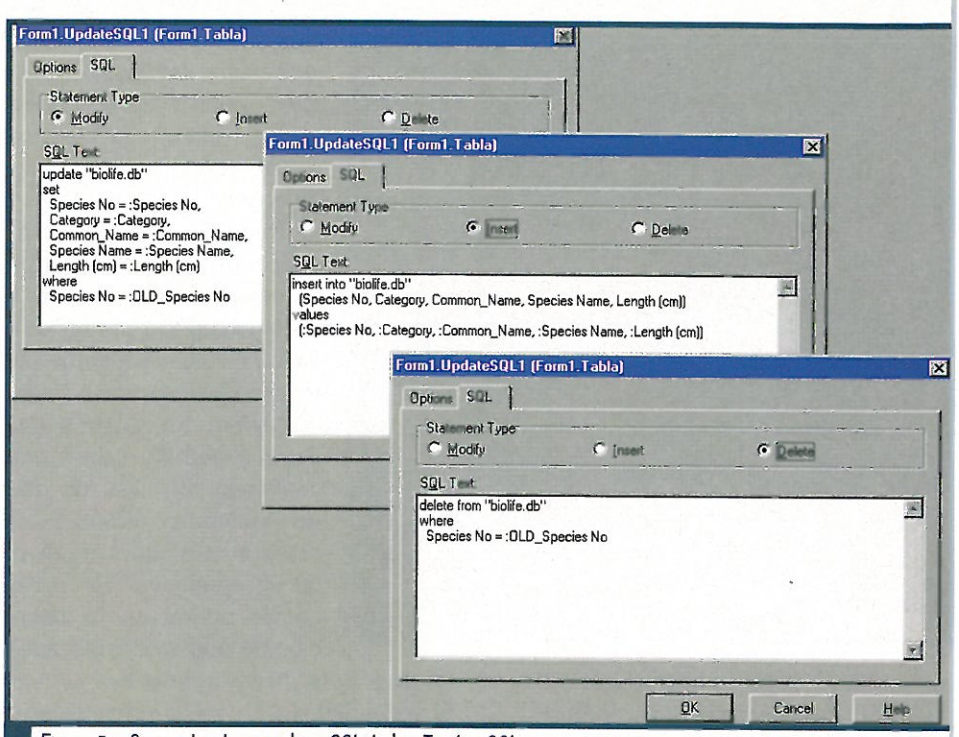


Figura 5. - Generador de consultas *SQL* de los *TupdateSQL.*

Es importante reducir la duración de las transacciones

Por ejemplo, el código del Listado 1 borraría primero los pedidos de un cliente, para eliminar después el cliente en sí. Es necesario hacerlo en este orden, para no violar la integridad referencial (no podemos borrar un cliente si de él cuelgan pedidos).

En realidad, el código no ejecuta específicamente un borrado. Más bien, lo que hace es indicar a los *TUpdateSQL* que ejecuten la sentencia asociada al tipo de actualización que se está realizando (indicado por la variable *UpdateKind).*

Nada nos impide usar otros mecanismos de actualización dentro del *OnUpdateRecord.* Podríamos, por ejemplo lanzar un procedimiento almacenado que actualice otras tablas además de las de clientes y pedidos.

En la ayuda de *Delphi* podéis consultar cómo generar las instrucciones *SQL* necesarias para actualizar las tablas, los parámetros disponibles, etcétera.

OTROS COMPONENTES

Hay algunos componentes más, que por su uso más esporádico, no hemos estudiado con detalle en la serie. Podéis ver un resumen de sus funcionalidades en la Tabla 2.

En esta serie, hemos dado una visión general del acceso a bases de datos desde *Delphi.* Para profundizar en el tema, recomiendo encarecidamente el libro "La cara oculta de *Delphi* 4" *(Ian Marteens, Ed. Danysoft).* En la *web* del autor, *www.marteens.com* podéis encontrar la versión para *C++ Builder* 4 en formato *PDF* (totalmente gratuita).

TABLA 2. Otros componentes relacionados con las BBDD

Componente	Descripción
TSession	Creación de sesiones de usuario. Puede controlar parámetros de la sesión, como la ruta del fichero de control de concurrencia usado en tablas *Paradox.*
TBatchMove	Movimientos masivos entre tablas, aun de diferentes *BBDD.* Util para copiar datos, tablas completas...
TNestedTable	Nuevas características de *Oracle 8*

¿TE FALTA ALGÚN NÚMERO DE...?

LA PRIMERA REVISTA DE PROGRAMACIÓN EN CASTELLANO

SÓLO PROGRAMADORES

ATENCIÓN

Los números no reseñados se encuentran AGOTADOS

Núm. 5. Procesador Pentium. Programación 3D. Curso de ensamblador. Técnicas de programación. Protoc de Comunicación. Tratamiento de imagen. Arquitectura. Técnicas de sonido. Sistemas abiertos. Proceso distribuido OS/2. Programación W. Curso C++.

Núm. 10. WWW. FallasParty'95. Curso ensamblador. Formatos gráficos. Programación básica. Herramientas gráficas. Comunicaciones tlf. Fivewin. Multitarea en protegido. VFoxpro 3.0. Linux. Progr de VGA. Redes loc. Progr Clipper. Sistemas abiertos. Curso C ++.

Núm. 13. VB Delphi. Programación. Curso Unix. Tratamiento digital de imagen. Cómo hacer una demo. Técnicas de sonido. Herramientas de progr con Threads. Progr 3D. Progr W. Grandes sistemas. Linux: TCP/IP. Programación Clipper. Listado BBS. Curso de C ++.

Núm. 20. WWW. Curso de programación. Curso de Unix. Cómo hacer
una demo. Grandes sistemas. Sistema operativo Linux. Creación de un compilador. Curso de VB 4.0. Programación de la VGA. Memoria expandida. Formatos gráficos. Visual C++ 4.0.

Núm. 22. WWW: Java. Curso de programación. Curso Unix. Grandes sistemas. Creación de un compilador. OSI. Modos X: la VGA. Linux: Distribución de memoria. VOC. El coprocesador 387. Cómo hacer una demo. Curso de VB 4.0: Eventos y métodos. CwinApp.

Núm. 25. Tecnología Web: CGI. Curso de programación. Grandes sistemas. Redes locales: TCP/IP. Curso de demos. Programación de la ROM BIOS. VGA: Resoluciones virtuales. Linux: Curses. Algoritmos Huffman. Curso de VB 4.0: Tipos de datos. Visual C++: Treectrl.

Núm. 26. HTML. Control del ratón. El protocolo IP. Utilidades MVS. Sistemas operativos de red. Euskal Party. Interfaces. Los Servicios Dos. Variables, constantes. Control de W95. El Gestor X-Window. VGA: Scroll X. Fundamentos de informática. Utilidades Shareware.

Núm. 27. Actualización dinámica. Programación del teclado. Detección de errores. Excepciones Delphi. SO de red. Sombreado Gouraud. El microprocesador. Funciones gráficas X. Programación GDI. Utilidades MVS. Microsoft e Internet. VB: Control de flujo.

Núm. 28. Metodología y transmisión de estado.Archivos indexados.Chequeo a Internet. Añadiendo gráficos a la demo. Protocolos de comunicaciones. Inteligencia artificial. Utilidades MVS. Optima 1.5. Programación del GDI. VB: Procedimientos.

Núm. 29. Programación Interactiva. Programación ADA. Compilador TMT Pascal. IPX/TK. Interfaces gráficas. Conmutación de paquetes. Intrabuilder prof. 1.0. VB: Objetos. Bitmaps independientes. Archivos. Inteligencia artificial. Servicios básicos. Distribuciones Linux.

Núm. 30. JavaScript: Chispa WEB. Lenguajes y herramientas. Mensajes TCP. CICS. Aplicaciones. VB: Menús, barras de herramientas. Borland C++ 5.01. IPX/SPX. La Iluminación. Programación de WSockets. ADA: Datos. El puerto serie. Programación multitarea.

Núm. 31. Web: Control de programa. Sistemas abiertos. Grandes sistemas. Netw Novell. Inteligencia artificial. Usuario Linux. VC++: OCX. ADA: Control. VB: Control de ratón y teclado. Programación de una demo. Programación TCL. Herramientas en el mercado.

Núm. 32. Protocolos Novell NW. Java: Primeros pasos. VB: OLE. C++ Builder. Aplicaciones matemáticas. Creación componentes (Delphi). Arquitectura Cliente/Servidor. Shockwave. Universal Server Informix. Herramientas abiertas TCL/TK. Criptografía. Intranet Linux.

Núm. 33. VB: Contenedor OLE. Objetos JavaScript. Java: Estructuras de control. Programación m/m. C ++ Builder y VCL. Rendimiento redes Novell. MMX, Hardware y programación. Propiedades y eventos Delphi. Layer3, MPEG rompe la barrera del sonido.

Núm. 34. JavaScript: Frames y Windows. Widgets TK. Intranet Linux. VB: Automatización OLE. Java: Constructores, composición, herencia. Programación avanzada C. Programación de Packet Driver. Criptografía. Novell NW. La herencia en el m.o.o.

Núm. 35. Liveconnect: puente entre Java y JavaScript. Criptografía de clave pública. BBDD en VB. Creación de componentes, trucos, y nociones importantes (Delphi). Interfaces, serialización y threads. Rad Magic 7. Xwindow. Servidores HP 9000.

Núm. 36. HTML dinámico con Netscape y JavaScript. Vamos a crear vida en un PC. Análisis del Pentium II. VB acceso a BBDD. Protección de datos. Programación Xwindow. Oracle 8. Programación de emuladores.Packages, clases anidadas y algún grano más.

Núm. 37. HTML dinámico con Netscape y JavaScript. Facturación en Novell NW. Funciones Hash. MD-5. Visión estereoscópica. Watcom C/C++ 11. VB BBDD. Algoritmos para programas de ajedrez. Vida artificial y algoritmos genéticos. Delphi 3.0. Directx y Directx SKD.

Núm. 38. El dominio dominado. Autentificación Kerberos. La función de evaluación. VC++ WCE. Transferencia de datos con DDX y DDV. Programación Xwindow. Aprendizaje automático y vida artificial. Entrada y salida (Java). Netcaster:tecnología Push,

Núm. 39. Netscape Netcaster y JavaScript 1.2. Intranet Linux. No es Internet todo lo que reluce. Quién es quién en Windows NT. ASN 1. Programación Xwindow. Movimiento selectivo y extensores. Atacando al efecto 2000. El lenguaje unificado para modelado.

Núm. 40. Generador de movimientos. Conexión a Internet con Linux. Programación de Directx con Delphi. Entorno de aplicaciones Forté. Visual JavaScript. Sockets y programación Internet con Delphi. Programación de objetos en VB. Perl. Sistemas Distribuidos.

Núm. 41. Jasmine. TCP/IP. Sistemas distribuidos. Depuración del uso de memoria dinámica en C. Visual Café. Mapas de mensajes.Merced, el futuro llega. IA: simbólico y subsimbólico. Programación Perl. Conectividad Windows-Unix con Samba.

Núm. 42. El método Simplex. Resolución de nombres de Internet. Tablas Hash. VB 5.0, los

documentos Activex. Sevicio FTP (Linux). Máquinas recreativas basadas en Intel. Softfactory 2000. Programación Directx con Delphi. Última versión IP. Modelado de objetos.

Núm. 43. Funciones hardware de los micros Intel. Cactus 3.0. El uso del tiempo. Computación evolutiva. Mapas de mensajes. Conectividad Windows-Unix con Samba. Programación de Directx con Delphi. Sistemas distribuidos. Simulación de sistemas.

Núm. 44. Creación de controles ActiveX. Programación de Directx con Delphi. Creación de un cliente hecho. Java servlets. Librería de acceso. Configuración clientes X con MWM. Correo electrónico (Linux). Control del Modem con VB. IIS. Integración numérica.

Núm. 46. Creación de controles ActiveX. Programación del acelerador gráfico Intel 740. BBDD relacionales con Post-gres. Programación de Directx (Delphi). Acceso remoto. Programación del modem. Creación de un servidor con Delphi.

Núm. 47. Programación de Directx (Delphi). Programación en Windows. Programación de WINCGI. SVGALIB. Arranque de Windows NT. Creación de controles ActiveX. Programación de servlets. Programación del modem. Representación gráfica de funciones 3D.

Núm. 48. Programación en Windows. Programación del API de Windows. Programación de Directx con Delphi. Reconocimiento de voz. Programación de WINCGI. SVGAL. Creación de componentes. Utilización de threads.

Núm. 52. AWT. Reconocimiento de voz (V). Glide. HTML dinámico (II). Herramientas para construir un cortafuegos (III). Creación de un buscador WEB. Interconexión de redes (II). Programación distribuida a objetos con Corba. WSH.

Núm. 53. HTML Dinámico. Reconocimiento de voz. Creación de un chat. Planificación de procesos (Linux). Glide. WSH. Visual Basic 6.0. Nuevas técnicas para bases de datos. Creación de un buscador WEB.

Núm. 54. Reconocimiento de voz. HTML dinámico. Creación de un buscador Web. Glide. Windows Scripting Host. VBScrip y Active Server Pages. Java: Tratamiento de imágenes. Estudio Práctico del TCP.

Núm. 55. HTML dinámico. Crear un banner. Glide. WSH. Java: Tratamiento de imágenes. Del HTML al acceso a BBDD. Desarrollo de aplicaciones con Videoconferencia. Novedades Java 2. Acceso a Oracle 8 desde C++ Builder. Creación de un buscador Web.

Núm. 56. Acceso a Oracle 8 desde C++ Builder. VB: Aplicaciones IIS). XML: El nuevo estándar de Internet. Glide. Multimedia Java. Del HTML al acceso a BBDD. Desarrollo de aplicaciones con Videoconferencia. VisualCafé 3.0. Directx 6.1. Creación buscador Web.

Núm. 57. Multimedia con Java: Sonido y Vídeo. Visual Basic: Aplicaciones IIS. Bases de datos: Desarrollo Cliente/Servidor. XML. El DOM de XML. Desarrollo de aplicaciones con Videoconferencia. Glide. Borland C++ Builder 4. Directx 6.1.

Núm. 58. Multimedia con Java. Captura de secuencias (Delphi). Redes: Transferencia de archivos. Desarrollo de aplicaciones con Videoconferencia. XML: XSL. BBDD: Desarrollo Cliente/ Servidor. Programación del registro de W. Directx 6.1. Visualage C++ 4.0.

Núm. 59. VB: Suite para Internet. Captura de secuencias (Delphi). Desarrollo de aplicaciones videoconferencia. XML: El lenguaje XS. Programación del registro de Windows. Redes: Transferencia de archivos. Directx 6.1. BBDD: Desarrollo Cliente/Servidor. Solaris 7.

Núm. 60. Tecnología ASP. Staroffice 5.1 para Solaris 7. Java: Encriptación. Captura de secuencia (Delphi). Linux: DNS. WNT 4.0 e IIS. Redes: Transferencia de archivos. Suite Internet. Correo electrónico. BBDD: Desarrollo C/S. Programación del registro de Windows.

Núm. 61. API de Telefonía. Programar un reproductor multimedia (Delphi). Linux: DNS a fondo. WNT 4.0 e IIS. Java: Encriptación. Visual Basic: Suite para Internet. La tecnología ASP. Bases de datos: Desarrollo Cliente/Servidor. Redes: Transferencia de archivos.

Núm. 62. Programación de Threads. API de Telefonía. Programar un reproductor m/m (Delphi). Novedades Delphi 5. El PC ha muerto, viva el PC. Linux: DNS a fondo. Tecnología ASP. Java: Encriptación. VB: Suite para Internet. Redes: Transferencia de archivos.

Núm. 63. Java Servlets y JDBC. Programación de Threads. Programar un reproductor multimedia con Delphi. API de Telefonía. Linux: DNS a fondo. La tecnología ASP (IV). Java: Encriptación. Visual Basic: Suite para Internet. Transferencia de archivos.

Núm. 64. Distribuciones Linux para programadores. Visual C++ y MFC. Java Servlet y JDBC. Microsoft Directx 7. Aplicaciones de Bases de datos en Delphi 5. Programación de Threads. Tecnología ASP. Java: Encriptación. Visual Basic: Suite para Internet.

Núm. 65. Los mejores componentes software del mundo. Aplicaciones de Bases de datos en Delphi 5. Linux: MP3. La tecnología ASP. Visual C++ y MFC. Firmado de Applets. Aplicaciones Web con acceso a BBDD basadas en Java.

Núm. 66. Curso de programación WAP. Firmado de Applets. Programación OpenGL (Delphi). VisualCafé 4. Linux: MP3. BBDD en Delphi 5. P.O.O: Las novedades de VB 7. Visual C++ y MFC. Aplicaciones Web con acceso a BBDD basadas en Java.

Núm. 67. Windows 2000 Server. Programación WAP. Programación OpenGL en Delphi. VisualCafé 4 Vs. Jbuilder 3. Linux: MP3. BBDD en Delphi 5. Programación orientada a objetos. Visual C++ y MFC. Aplicaciones Web con acceso a BBDD basadas en Java.

Núm. 68. Windows2000 Server contra Linux. Progr WAP. Progr OpenGl (Delphi). P.O.O. C++ Builder 5. Aplicaciónes de bases de datos con Delphi 5. Jbuilder 3.5. VC++ y MFC. Aplicaciones Web con acceso a BBDD Java.

Núm. 69. XML & Java. Programación WAP. Fastlib 2000. P.O.O. Optimizeit contra Jprobe. Aplicaciones BBDD Delphi 5. Configuración de X-Window. VC++ y MFC. Progr OpenGL en Delphi. Adquisición de datos con C++ y Osciloscopio Virtual.

BOLETÍN DE PEDIDO

Rellene o fotocopie el cupón y envíelo a REVISTAS PROFESIONALES, S.L.

(Revista SÓLO PROGRAMADORES). C/ San Sotero, 5 - 1ª Planta

Tel.: 91 304 87 64 • Fax: 91 327 13 03 • 28037 MADRID • http:www.revistasprofesionales.com

Deseo me envíen los número/s .....

NOMBRE Y APELLIDOS .....

EDAD ..... PROFESIÓN ..... TELÉFONO .....

DOMICILIO .....

CIUDAD ..... PROVINCIA ..... CÓDIGO POSTAL .....

FORMAS DE PAGO

☐ Giro postal a nombre de REVISTAS PROFESIONALES, S. L. ☐ Talón bancario a nombre de REVISTAS PROFESIONALES, S. L.

☐ Domiciliación bancaria ☐ Contra reembolso

DATOS DE DOMICILIACIÓN:

Banco .....

Domicilio .....

Número de cuenta ____/____/__/__________

Titular .....

Tarjeta VISA

____/____/____/____ Fecha caducidad: .....

Fecha:

Firma,

Precio por unidad: 950 pesetas + 700 pesetas de gastos por cada envío (hasta un máximo de 10 revistas).

Páginas web dinámicas con el lenguaje de scripts PHP

Jacinto Mata Vázquez. *Profesor de lenguaje y sistemas informáticos de la UHU.*

En este artículo vamos a descubrir la potencia de uno de los lenguajes de *scripts* con mayor protagonismo en los últimos tiempos gracias a su capacidad de crear páginas *Web* que interactúan con bases de datos.

INTRODUCCIÓN

as páginas *web* son un medio de información cada vez más utilizado particularmente y en la empresa. La evolución de la tecnología *web* se demuestra n las continuas revisiones del lenguaje *TML*, con la inclusión de aspectos dámicos, hojas de estilo, etcétera.

Por otro lado, hay herramientas para otar a las páginas *web* de un carácter inámico e interactivo, además del de iente de información. La forma habiial de realizar esta comunicación es iediante *scripts* programables de diversas maneras. Entre las más utilizadas, stán el lenguaje *JavaScript*, los *CGI* escritos en *Perl* o en *C*, y el lenguaje *PHP*, ue está cobrando gran protagonismo.

En síntesis, podemos definir *PHP* omo un lenguaje de *scripts* incrustado n *HTML*, es decir, que podemos mezar en el mismo fichero, código *HTML* stándar con *scripts* escritos en *PHP*. na de las ventajas de *PHP* es su sintais simple, que permite escribir *scripts* otentes con muy pocas líneas de códio. Otra ventaja es que su código, a diferencia de otros lenguajes, se ejecuta n el servidor, por lo que, además de ser ás rápido, el cliente sólo recibe el reultado de dicha ejecución, quedando culto el diseño de nuestros programas.

Su característica más significativa es la facilidad para realizar aplicaciones *web* que interactúan con bases de datos. *PHP* posee funciones para trabajar con la mayoría de los sistemas gestores de bases de datos, así que si quieres gestionar tu información mediante una base de datos accesible con tus páginas *web* bajo *Linux*, la mejor elección es *PHP* junto al *SGBD* preferido.

INSTALACIÓN

Naturalmente, debemos tener un servidor *http* con el que gestionar nuestras páginas, bases de datos, *scripts*, aplicaciones *CGI*, etcétera. En nuestro caso, usaremos el *Apache*, por dos razones: por ser el más extendido en la actualidad en la plataforma *Linux*, y porque *PHP* es más eficiente cuando se instala como un módulo de *Apache*.

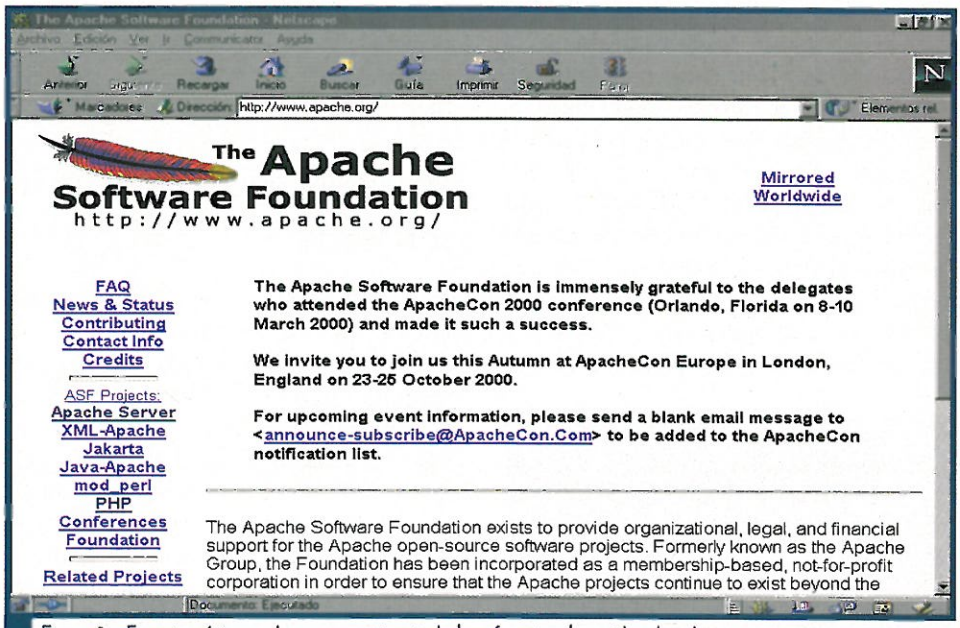


Figura 1. - En esta página podemos encontrar todo lo referente al servidor *Apache*.

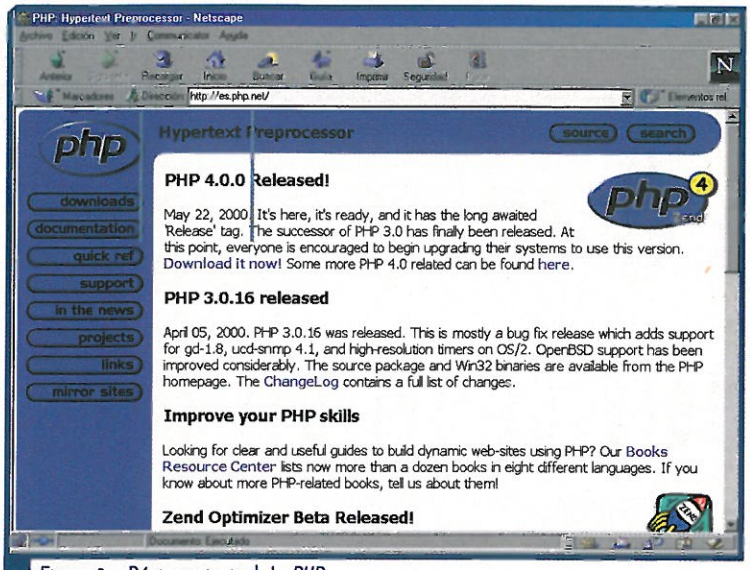


Figura 2. - Página principal de *PHP*.

Lo primero, es conseguir la última versión de *PHP3* para *Linux*: en la página *PHP* (*www.php.net*) encontraremos las últimas versiones.

Buscaremos un lugar donde descomprimirlo, por ejemplo el directorio */usr/local*, y ejecutaremos *tar xvfz php-3.0.16.tar.gz*

Esto creará el directorio */usr/local/php-3.0.16*. Una vez dentro del mismo, hay que preparar el fichero de compilación. Hay 2 formas de hacerlo...

MÉTODO 1

Consiste en ejecutar directamente el *script configure* con el conjunto de opciones que deseamos personalizar (para ver todas las opciones, teclearemos *./configure —help*). Aunque existen otras formas de instalación, nosotros usaremos la instalación de *PHP3* como un módulo de *Apache*, por las razones indicadas más arriba. Para ello, ejecutaremos el comando *./configure —with-apache = directorio base del Apache*, donde *"directorio base del Apache"* es el directorio donde se encuentra el código fuente del servidor. Si no ha habido ningún problema, podemos proceder a la compilación y a la instalación de los ejecutables mediante los comandos *make* y *make install*.

Si el proceso ha sido correcto, ya tenemos el lenguaje instalado en nuestro equipo, y se habrán copiado los módulos de *PHP3* en el directorio base de *Apache*. Sólo queda preparar nuestro servidor *web* para que pueda ejecutar los *scripts* escritos en *PHP*. Lo mejor es recompilar el servidor con las nuevas opciones, para lo que desde el directorio base de *Apache* ejecutaremos los comandos:

```
./configure —prefix=/dir de instalación —activate-module=src/modules/php3/libphp3.a
make
make install
```

Donde *dir de instalación* es el directorio donde se ubicarán los ejecutables de *Apache*. Después de esto, tendremos un nuevo ejecutable *httpd* con soporte para *PHP3*. Sólo falta un pequeño detalle para que todo funcione correctamente, y es que tenemos que indicar a nuestro servidor cuál será la extensión de los ficheros escritos en *PHP3* para que pueda reconocerlos. Para hacer esto editamos el fichero *conf/srm.conf* o *conf/httpd.conf* (para las nuevas versiones de *Apache*) y dejamos sin comentar las líneas:

```
AddType application/x-httpd-php3 .php3
AddType application/x-httpd-php3-source .phps
```

TABLA 1. SGBD soportados por *PHP*

Adabas D	*MySQL*	*FilePro*
dBase	*Solid*	*Informix*
InterBase	*Sybase*	*Oracle*
mSQL	*Velocis*	*PostgresSQL*

Los scripts de PHP se ejecutan en el servidor

Con la primera línea, indicamos que la extensión de los ficheros *PHP3* será *.php3*. Esta es la opción que trae por defecto, y la más recomendable, pero si algún lector quiere utilizar otra extensión, este es el lugar donde hacerlo. La segunda línea sirve para indicar que los ficheros con extensión *.phps* se visualicen con la palabras reservadas del lenguaje en diferentes colores en lugar de ejecutarse. Esto puede ayudarnos a depurar nuestros códigos.

Por último, en el directorio */usr/local/php-3.0.16* se habrá creado el fichero de configuración *php3.ini-dist*, que copiaremos con el nombre de *php3.ini* en el directorio */usr/local/lib*.

Esta es la ubicación por defecto, pero si queremos una ubicación distinta usaremos la directriz *—with-config-file path=/nueva ubicación* en la línea de comandos del *script* de configuración. En este fichero se guardan las opciones de configuración con las que se ejecutan nuestros *scripts PHP*, así que podemos editarlo y modificar alguna de las directrices. Este fichero se lee cada vez que arranca el servidor *web*, de manera que si realizamos algún cambio en él, debemos volver a lanzar nuestro servidor para que dichos cambios sean efectivos.

MÉTODO 2

Las últimas versiones de *PHP* proporcionan un *script* de configuración en el que se pregunta al usuario sus preferencias de instalación. Así no tenemos que poner todas las opciones de configuración en el momento de ejecutar el *script configure* ya que son muchas y sería imposible conocer todas sus directrices. La forma de lanzarlo es con el comando *./setup*

Al acabar este *script*, se ejecuta *configure* con las opciones elegidas anteriormente. Además, se creará un fichero llamado *do-conf*, en el que están las opciones que hemos elegido en el proceso de configuración. Podemos editarlo, y eliminar, añadir o modificar alguna opción sin tener que ejecutar de nuevo el fichero *setup*. Simplemente ejecutaremos el fichero *./do-conf*.

Después de este proceso, volveremos a compilar y a instalar el programa según el método 1. Una vez instalado el programa, recuerda que hay que lanzar de nuevo el servidor *web* para que recoja los cambios.

PROBANDO EL PRIMER SCRIPT

Comprobemos si la instalación de *PHP3* se ha sido correcta: como es habitual en el comienzo del aprendizaje de un nuevo lenguaje, crearemos un *script* que muestre un mensaje por pantalla (puede ser el famoso ¡Hola mundo!). Para escribirlo, usaremos nuestro editor favorito sin olvida

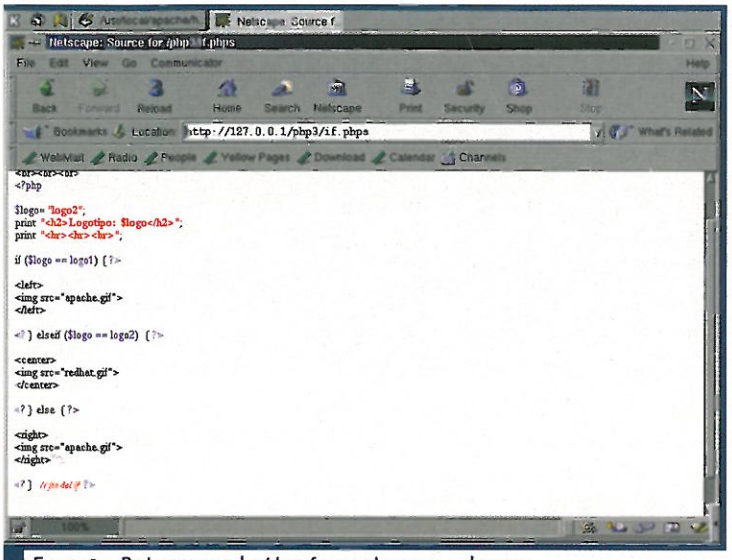


Figura 3. - Podemos ver el código fuente de manera elegante.

nombrar al fichero con la extensión que hayamos elegido en la configuración (*.php3* o *.phtml*).

```
<html>
<body>
<center><h1>Mi primer script en
      PHP3</h1></center>

<?php
print "<BR><BR>";
print "<center><h2>¡¡¡ Enhorabuena
      !!!</h2></center>"; //felicitación
print "<BR><BR>";
print "<h3>Si puedes leer esto, es que PHP3 está
      bien instalado</h3>";
* si sale algún mensaje de error
e debe repasar el proceso de instalación */
>
</body>
</html>
```

Sitúa el fichero en la ruta correspondiente a las páginas *web* y conéctate a tu servidor con el navegador para comprobar el resultado. Si en pantalla aparece algún mensaje de error o aparece el código al cual, repasa los pasos de la instalación para ver si has olvidado alguna acción.

PHP tiene funciones para la mayoría de los sistemas gestores

Antes de analizar algo de la sintaxis, mira el código fuente de la página. Podrás observar que no aparece nada del código *PHP* que escribiste. Esto se debe al propio funcionamiento de los *scripts* incrustados en el servidor, esto es, el código *PHP* es examinado y procesado en el servidor y devuelve al cliente, únicamente, el código *html* generado.

En este código podemos observar algunos aspectos interesantes: el código *HTML* necesario para mostrar la página *web* y el código *PHP*, conviven en el mismo fichero. Para distinguirlos, se necesitan unos delimitadores. Así, cualquier bloque de código que esté delimitado por las etiquetas *<?php* y *?>* será procesado como código *PHP*. Uno de los aspectos más interesantes de este lenguaje es que podemos colocar el código en cualquier parte del fichero.

Hay otras formas de delimitar el código *PHP*, que tendrán validez si la hemos especificado en el momento de la configuración. La forma más cómoda de crear un bloque con código *PHP* es usar los delimitadores *<?* y *?>*, pero para eso tendrá que estar habilitada la opción *short-tags* en el fichero *php3.ini*.

Otro detalle es el punto y coma del final de la línea. Este es el símbolo, como en la mayoría de los lenguajes de programación, que indica un final de instrucción. Además, podemos comentar nuestro código de diversas formas, ya que el lenguaje *PHP* soporta el estilo de los comentarios de C, C++ y de la *shell* de *UNIX*.

SINTAXIS: TIPOS Y VARIABLES

En *PHP* no hay que declarar las variables y asignarlas un tipo antes de su uso, pues esa operación se hace en tiempo de ejecución. Significa que una misma variable puede adoptar diferentes tipos según el contexto. Por ejemplo, si le asignamos a la variable *mivar* una cadena de caracteres, esa variable será de tipo cadena; pero si más adelante, asignamos a *mivar* un valor entero, entonces la variable pasa a ser de tipo entero. Para asignar un valor a una variable, sólo hay que anteponer el símbolo dólar ($) al nombre de la variable:

```
$mivar = 777;
$mivar = "ba tiene tres pelos";
```

Como dijimos, la conversión de tipos en *PHP* es automática, aunque se puede forzar explícitamente el tipo de una variable con lo que se conoce como *type casting*, anteponiendo a la variable el nuevo tipo en el que se quiere transformar.

```
<?php
$a = 25;                                  // la va-
      riable a es de tipo entero
$b = 10.58;                    // la variable b es
      de tipo real
$suma = $a + $b;               // la variable suma
      es de tipo real
$suma = $a + (int)$b; // la variable suma es de tipo
      entero
print "La suma de $a + $b es $suma";
?>
```

La salida de este programa será: *La suma de 25 y 10.58 es 35*. Como se puede apreciar, el tipo de la variable *b* no se ha visto afectado, ya que este tipo de conversión sólo influye en cómo son evaluadas las variables, pero sin cambiar su tipo.

El cliente sólo ve código HTML al recibir la pagina

Dentro de los tipos descritos en la Tabla 2, requieren una especial atención los *arrays*, diferentes a los de otros lenguajes de programación. Pueden ser unidimensionales o multidimensionales y no tenemos que declarar su tamaño con anterioridad para poder usarlos. Por otra parte, sus índices pueden ser números o cadenas de caracteres, y además, se permiten *arrays* con valores de distintos tipos. Veamos un ejemplo:

```
<?php
$personas = array (
      "p1" => array (
            "nombre" => "Alfonso",
            "edad" => 25,
            "lugar" => "Madrid"),
      "p2" => array (
            "nombre" => "Julián",
            "edad" => 36,
            "lugar" => "Córdoba"),
      "p3" => array (
            "nombre" => "Ana",
```

TABLA 2 . Tipos de datos soportados por *PHP*.

enteros	valores enteros	*$a = 65* *$b = -28*
reales	valores con decimales	*$a = 250.25* *$b = 2.5e3*
cadenas	combinaciones de caracteres delimitados por comillas dobles	*$c = "Introduce un valor"*
arrays	admite tanto *arrays* indexados como *arrays* asociativos	*$a[] = "Hola"* *$b["lunes"] ="examen"*
objetos	admite la utilización de clases	*$a = new clase*

```
        "edad" => 17,
        "lugar" => "Cádiz"));

for (reset($personas); $c1=key ($personas); next
    ($personas))
for (reset($personas[$c1]); $c2=key($personas[$c1]);
    next($personas[$c1]))
        print "personas[$c1][$c2] = " .$perso-
    nas[$c1][$c2] . "<BR>";
?>
```

Este programa crea un *array* asociativo bidimensional y lo carga con algunos valores, para después mostrarlos por pantalla. No te preocupes si no entiendes el significado de la sentencia *for*. En secciones posteriores se explica el funcionamiento de las estructuras de control.

PHP, además de tener implementadas una serie de constantes, también permite definir otras nuevas mediante la función *define()*.

```
<?php
define ("PIE", "<BR> (c) 2000. Página escrita en
    PHP3");
print PIE;
?>
```

OPERADORES

Los operadores de *PHP*, en su mayoría, trabajan y utilizan los mismos símbolos que los del lenguaje *C*. El único que se diferencia es el operador de concatenación de cadenas, para el que el *PHP* utiliza el punto ".". En las siguientes líneas vemos el funcionamiento de algunos operadores:

```
<?php
$v1 = 25;              // $v1 vale 25
$v1++;                 // $v1 vale 26
$v1 *= 10;             // $v1 vale 260
$v1 = ($v2 = 30) - 10; // $v1 vale 20 y $v2 vale 30
$c1 = "Me gusta ";
$c2 = "PHP3";
print $c1 . $c2;       // muestra por pantalla "Me
    gusta PHP3"
$c1 .= $c2;            // en $c1 se almacena el texto
    "Me gusta PHP3"
?>
```

ESTRUCTURAS DE CONTROL

De la misma forma que los operadores, las estructuras de control de *PHP* son similares a las del lenguaje *C*, como podemos ver en la Tabla 4.

No es necesario declarar las variables antes de usarlas

Además, podemos utilizar las estructuras de control para manipular el resto del código *HTML* de nuestro fichero. Así podemos conseguir páginas que muestren textos diferentes según el valor de una variable, o que muestren una imagen tantas veces como se le indique en un bucle:

```
<?php
$logo = "logo2";
print "<h2>Logotipo: $logo</h2>";
print "<br><hr><br>";
if ($logo == logo1) { ?>

<left>
<img src="apache.gif">
</left>

<? } elseif ($logo == logo2) { ?>

<center>
<img src="redhat.gif">
</center>

<? } else { ?>

<right>
<img src="php.gif">
</right>

<? } // fin del if ?>
```

Como se puede ver, las secciones ajenas al código *PHP* están en la estructura de control *if*. Esto hará que en el navegador aparezca una de las tres imágenes dependiendo del valor de la variable *logo*.

TABLA 3 .Funciones de manipulación de *arrays*.

array array (...)	crea un nuevo *array*
elem current (array a)	devuelve el elemento actual de un *array*
array each (array a)	devuelve el par clave-valor actual de un *array*
end (array a)	sitúa el puntero interno al final del *array*
elem key (array a)	devuelve el índice de la posición actual del *array*
int ksort (array a)	ordena un *array* por clave
elem next (array a)	avanza el puntero interno de un *array*
elem prev (array a)	retrocede el puntero interno de un *array*
elem reset (array a)	sitúa el puntero interno en la primera posición del *array*
int sizeof (array a)	devuelve el número de elementos de un *array*

FUNCIONES

En *PHP* las funciones tienen que definirse antes de ser referenciadas.

```
<?php
function potencia($x,$n) {
```

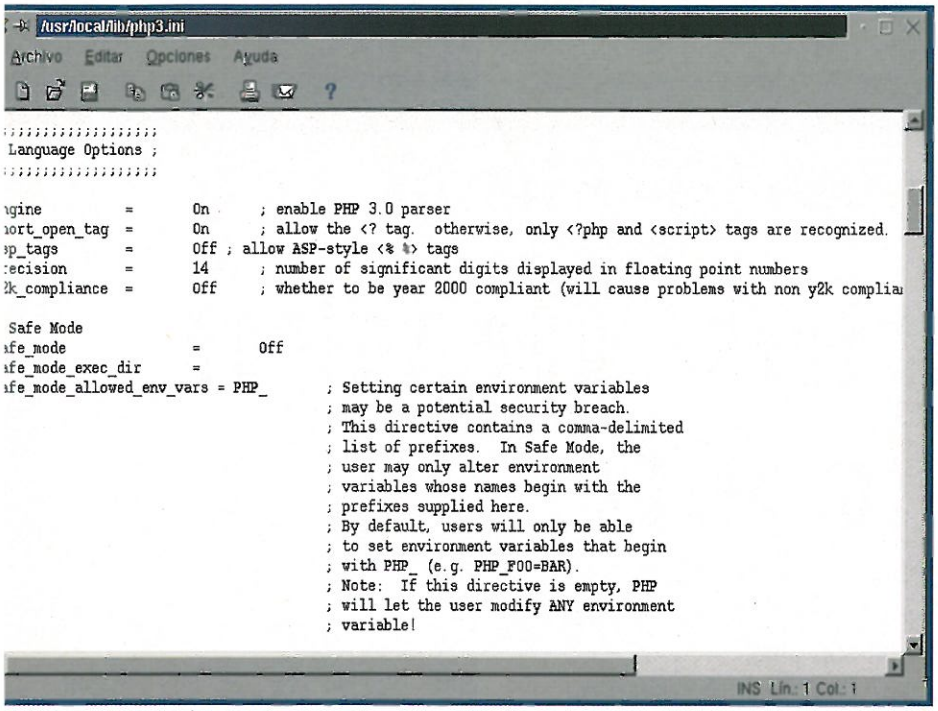


Figura 4. - Sección del fichero *php3.ini*.

```
r($p=1; $n>0; $n—) $p = $p*$x;
turn($p);


 = 2; $n = 4;
 = potencia($x,$n);
nt "$x elevado a $n es %$y ";
```

Además, las funciones pueden devolr un valor o un conjunto de valores.

```
php
ction precios($prod) {
iva=$prod[2] * 0.16;
total=$prod[2] + $iva;
turn array($prod[2], $iva, $total);


l[0]="cafetera";
l[1]="acero inoxidable";
l[2]=15000;

($precio, $iva, $total) = precios($p1);
nt "precio: $precio <BR>";
nt "IVA: $iva <BR>";
nt "Precio total: $total <BR>";
```

Por otro lado, si no se especifica lo contrario, los argumentos se pasan a las funciones por valor, es decir, que los cambios hechos sólo tienen efecto dentro de la función. Si queremos pasar un argumento por referencia, sólo tenemos que anteponerle el carácter *ampersand* (&).

```
<?php
function en_euros(&$p) {
 $p .= " euros";
}

$precio=24.2;
en_euros($precio); // la variable precio es "24.2 eu-
      ros"
?>
```

TABLA 4. PHP soporta diferentes estructuras de control.

if	*while*	*break*
else	*do..while*	*continue*
elseif	*for*	*switch*

LAS FUNCIONES INCLUDE() Y REQUIRE()

Hay dos funciones que nos pueden ahorrar muchas líneas de código cuando estemos diseñando nuestras páginas. La función *require()* importa el fichero especificado en cuanto aparece por primera vez en cualquier lugar del *script*, es decir, que si la ponemos dentro de una estructura condicional, el fichero será importado incluso si no se cumple la condición. Sin embargo, la función *include()* importa el fichero especificado cada vez que se encuentra en el *script*, pero en tiempo de ejecución. Por tanto, si lo que queremos es importar un fichero según de una determinada condición, usaremos la función *include()*.

En general, usaremos la función *require()* de la misma forma que la directriz *#include* en *C*, es decir, cuando queremos usar una serie de funciones que se encuentran definidas en un fichero independiente, mientras que la función *include()* se utilizará para crear plantillas, o sea, para insertar secciones completas de código en zonas determinadas de nuestro *script* principal. Suele ser habitual asignarle la extensión *.inc* a los ficheros de importación para distinguirlos de los propios *scripts PHP*.

Más abajo podemos ver el código de los ficheros de cabecera:

```
<html>
<head>
<title>
<?php print $titulo ?>
</title>
</head>
<body>
<center><h1><?php print $presentacion
    ?></h1></center>
<hr>
<br>
```

Y para el pie:

```
<br>
<hr>
(c) 2000. Página escrita en PHP3
<br>
<center><img src = logotipo.gif"></center>
</html>
```

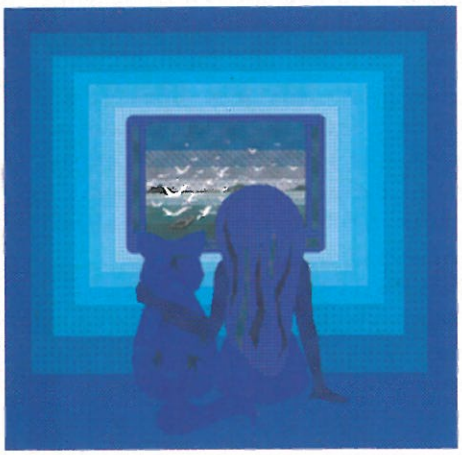
Un *script* que utiliza estos ficheros:

```
<?php
$titulo = "Página principal";
$presentacion = "¡¡¡ Bienvenidos !!!";
include "cabecera.inc";
?>
Aquí va el contenido de la página
<?php
include "pie.inc";
?>
```

CONCLUSIÓN

Los ejemplos descritos han utilizado variables que han sido cargadas con valores dentro del propio código. Pero si nuestro objetivo es crear páginas dinámicas en las que se pueda interactuar con el usuario, debemos conocer los mecanismos de *PHP* para recoger información desde los navegadores mediante un formulario, procesarla, y devolver una respuesta al usuario en forma de página *web*. Esto lo veremos en el próximo número.

Programación OpenGL en Dephi (V)



Texturas que saltan a la vista

Juan Luis Ceada Ramos.
Diplomado en Informática de Sistemas.
Emilio Nestal Díaz.
Ingeniero Técnico en Informática.

Vamos a aprender a usar las texturas para mejorar el aspecto gráfico de nuestras escenas sin tener que generar un número desmesurado de polígonos.

TEXTURAS

Una textura es una imagen bidimensional que aplicada sobre un polígono, se utiliza para dar realismo a las escenas sin tener que cargarlas de polígonos, reduciendo así el tiempo de cálculo. Su uso aumenta el requerimiento de memoria, por lo que deberemos aceptar una velocidad de ejecución más baja. Cualquiera puede diseñar sus propias texturas. Basta con crearlas con cualquier programa de diseño gráfico, y grabarlas en algún formato conocido. La única limitación es que sus dimensiones deben ser potencia de dos. Por ejemplo 128 x 128, 32 x 32 o 32 x 512 (no tiene por qué ser cuadradas). Las texturas s pueden almacenar en cualquier format gráfico, ya que *OpenGL* no trabajará d rectamente con los ficheros, sino que l tendremos que pasar en la mayoría d los casos un puntero a la zona de me moria de datos en formato *RGE OpenGL* trabaja con varios tipos de tex turas: de una dimensión (*1D*), de dos d mensiones (*2D*) y multimapa.

Figura 1. - Sin las texturas, juegos como *Half-line* no existirían.

1D Y 2D

Una textura *1D* es una imagen con an chura o altura de valor *1*. Son útile para realizar degradados de color.Par crearlas *1D* usaremos el siguiente pro cedimiento: *glTexImage1D (targe TGLEnum; level, internalforma TGLint; width: TGLsizei; borde TGLint; format, atype: TGLEnum; p xels: Pointer);*

El primer argumento (*target*) indic el tipo de textura que generaremos, y s valor debe ser *GL_TEXTURE_1D*. *L vel* indica la cantidad de detalle, y no malmente es *0* (a menos que usemo texturas multimapa). *InternalForm*

dica el número de *bytes* que definen un xel (1 para color indexado, 3 para GB, 4 para *RGBA*).

Width y *Border* especifican el tamaño. borde (*Border*) puede tener 0 ,1 o 2 píles de ancho. *Width* indica la anchura el borde, y debe ser potencia de 2.

El parámetro *Format* indica cómo se presentan los colores (componentes GB, luminancia, etc.) Los valores que ede tomar son: *GL_RGB*, *GL_RGBA*, _LUMINANCE y *Gl_COLOR_INDEX*. Como es habitual, trabajaremos n el formato *GL_RGB* (para trabajar n texturas de 24 *bits*). El parámetro *pe* permite indicar cómo se almacenan píxeles de la textura, y puede tomar siguientes valores: *GL_UNSIGNED TE, GL_BYTE, GL_SHORT*. Normalmente trabajaremos con *GL_UNGNED_BYTE* (cada componente GB está representada por un número tre el 0 y el 255).

Por último, con el parámetro *Pixels*, licaremos la posición de memoria en la e se almacenan los datos de la imagen.

RÁMETROS

os parámetros pueden modificarse antes de la creación de una textura, y ctan a la calidad y velocidad de dibujo nuestras escenas. Para definirlos, amos la función *glTextParameter (tar-, pname: TGLEnum; param: TGLint)*. El primer parámetro indica el tipo de textura al que afectará el cambio (a las *1D*, a las *2D* o a ambas a la vez). El segundo, indica qué parámetro se modifica. Los posibles valores son: *GL_TEXTURE_MIN_FILTER* (filtro usados a la hora de reducir la imagen), *GL_TEXTURE_MAX_FILTER* (al ampliarla), *GL_BORDER_COLOR* (color del borde), *GL_TEXTURE_WRAP_S* (especifica qué hacer con la textura si es más pequeña que el polígono donde va situada) y *GL_TEXTURE_WRAP_T* (especifica qué hacer con la textura si es más grande que su polígono).

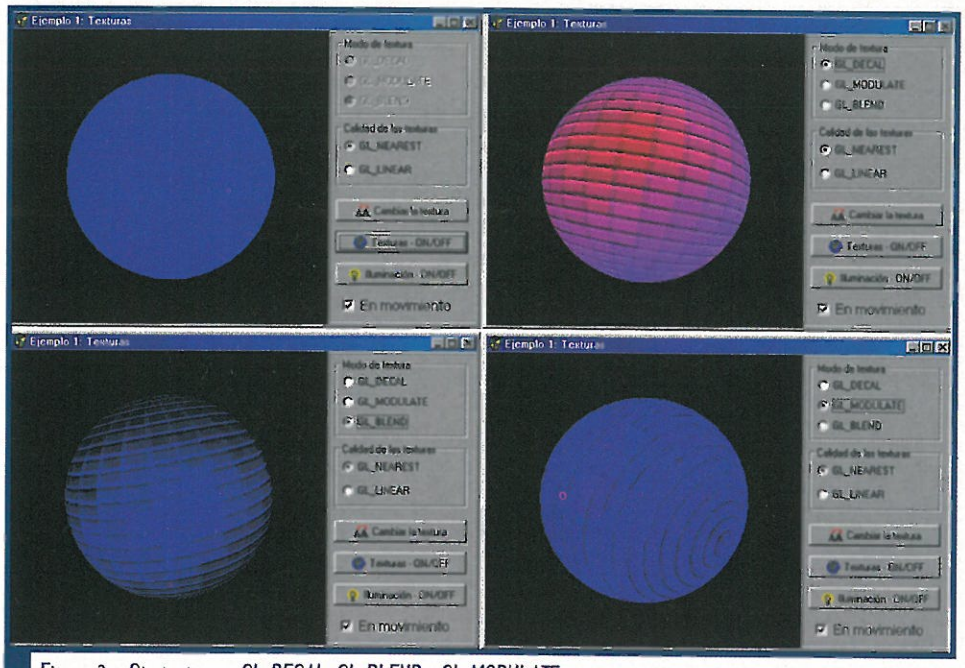


Figura 3. - Sin texturas, *GL_DECAL*, *GL_BLEND* y *GL_MODULATE*.

Según lo que queramos cambiar, *param* recibe unos valores u otros. Si *pname* es igual a *GL_TEXTURE_MIN_FILTER*, *param* puede valer *GL_NEAREST* (vecino mas próximo), *GL_LINEAR* (interpolación lineal), *GL_NEAREST_MIPMAP_NEAREST*, *GL_NEAREST_MIPMAP_LINEAR*, *GL_LINEAR_MIPMAP_NEAREST* y *GL_LINEAR_MIPMAP_LINEAR* (veremos el significado de estos últimos parámetros más adelante). Con las llamadas a la función *glTextParameter(GL_TEXTURE_1D, GL_TEXTURE_MIN_FILTER, GL_LINEAR)* y *glTextParameter (GL_TEXTURE_1D, GL_TEXTURE_MAX_FILTER, GL_LINEAR)*, indicamos a *OpenGL* que si necesita reducir o aumentar el tamaño de textura, lo haga aplicando un filtro de tipo *GL_LINEAR* (utiliza interpolación lineal, suavizando la imagen). Estos filtros consumen más tiempo de cómputo que los filtros *GL_NEAREST*, pero los resultados son mejores (ver Figura 2).

En caso de que el valor de *pname* sea *GL_TEXTURE_MAX_FILTER*, *param* puede tomar los valores *GL_NEAREST* y *GL_LINEAR*. Para *GL_TEXTURE_WRAP_S*, *param* puede valer *GL_REPEAT* (que indica que si la textura es más pequeña que el polígono,

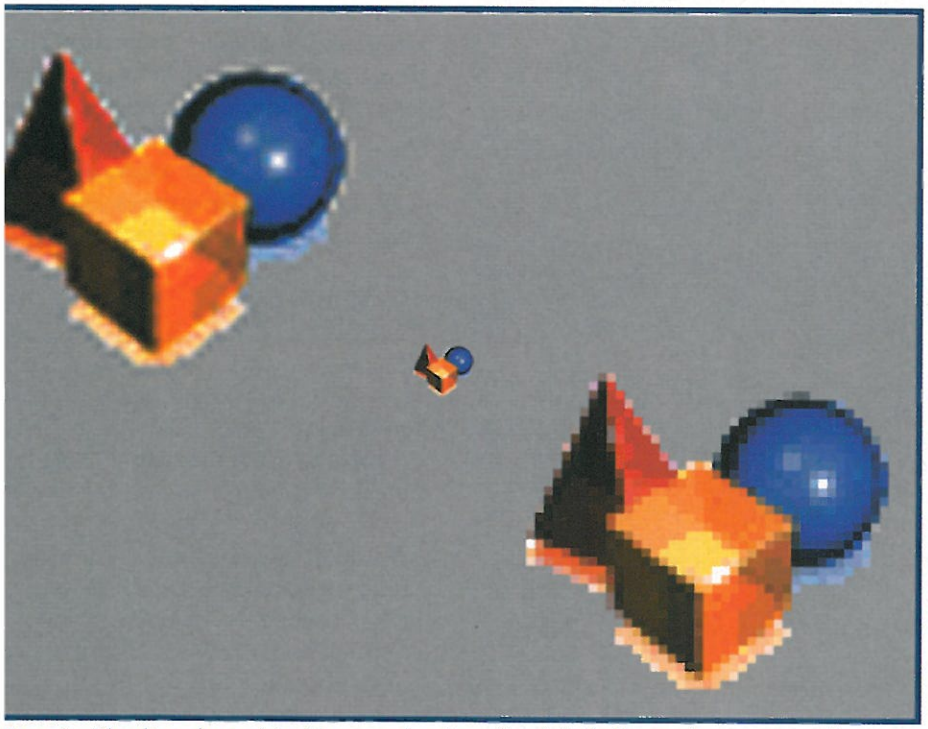


Figura 2. - Filtro linear (izquierda) y del vecino más próximo (derecha). En el centro, la imagen original.

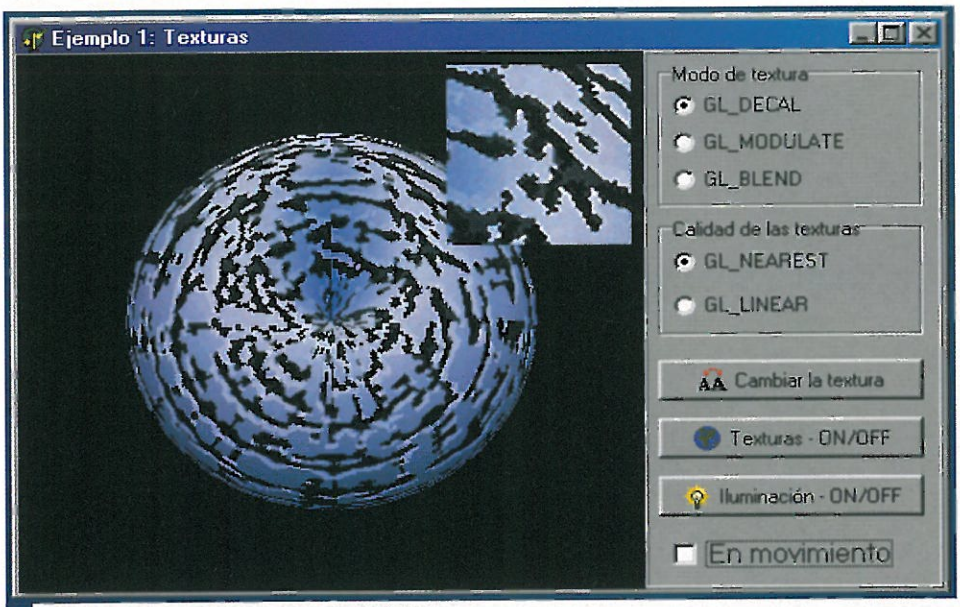


Figura 4. - Uso de un filtro de tipo *GL_NEAREST*.

ésta debe repetirse por toda la superficie del mismo, a modo de mosaico), o *GL_CLAMP* (usa el color del borde para rellenar las zonas del polígono que la textura no cubre).

TEXTURAS 2D

Una textura *2D* se usa para dar realismo a las imágenes, ahorrando polígonos. Esta función permite definir una textura *2D*:

```
Procedure glTexImage2D (target: TGLEnum; level, internalformat: TGLint; width, height: TGLsizei;
border: TGLint; format, atype: TGLEnum; Pixels:Pointer)
```

Una textura es una imagen superpuesta a los polígonos

El significado de los parámetros es igual que en las texturas *1D* (excepto que en este caso necesitamos indicar con *height* la altura de la textura).

EJEMPLO 1: Función que carga las texturas en memoria

```
procedure TFPrincipal.FormCreate(Sender: TObject);
var
  Header: Pointer;
  HeaderSize: DWord;
  ResultB:Boolean;
begin
  Bitmap:=TBitMap.Create;
  Bitmap.LoadFromFile(ExtractFilePath
    (Application.ExeName)+ 'Textura_1.bmp');
  GetDIBSizes(Bitmap.Handle, HeaderSize, BitsSize);
  CompruebaImagen(Bitmap);
  ArreglaImagen(Bitmap);
  Header := AllocMem(HeaderSize);
  GetMem(Puntero1,BitsSize);
  ResultB:=GetDIB(Bitmap.Handle, Bitmap.Palette, Header^,
    Puntero1^);
  Bitmap.LoadFromFile(ExtractFilePath(
Application.ExeName)+'Textura_2.bmp');
  GetDIBSizes(Bitmap.Handle, HeaderSize, BitsSize);
  CompruebaImagen(Bitmap);
  ArreglaImagen(Bitmap);
  Header := AllocMem(HeaderSize);
  GetMem(Puntero2,BitsSize);
  ResultB:=GetDIB(Bitmap.Handle, Bitmap.Palette, Header^,
    Puntero2^);
  FreeMem(Header, HeaderSize);
  Bitmap.free;
end;
```

DIBUJO DE POLÍGONOS

Como en otras ocasiones, a menos que activemos explícitamente el uso de texturas, no notaremos cambios en nuestras escenas. Para ello, hay que recurrir nuevamente a las funciones *gl_enable* y *gl_disable*. Este ejemplo, activa las texturas *2D* y desactiva las *1D*:

```
Gl_disable(GL_TEXTURE_1D);
Gl_enable (GL_TEXTURE_2D);
Gl_textEnvi(GL_TEXTURE_ENV,
      GL_TEXTURE_ENV_MODE, GL_DECAL)
```

Con la función *gl_TextEnv* se controla la aplicación de las texturas sobre los polígonos. El primer parámetro siempre es *GL_TEXTURE_ENV*. El segundo indica qué parámetro se modifica (*GL_TEXTURE_ENV_MODE*, la forma de dibujar la textura o *GL_TEXTURE_ENV_COLOR*, color para las mezclas). El tercero, cuando el segundo es *GL_TEXTURE_ENV_COLOR*, es un color *RGBA*. Cuando el segundo es *GL_TEXTURE_ENV_MODO*, el tercero puede valer *GL_DECAL* (la textura se copia directamente sobre polígono), *GL_BLEND* (la textura se mezcla con un color constante, antes de colocarse sobre el polígono) o *GL_MODULATE* (la textura se multiplica por el contenido de la superficie sobre la que se va a depositar antes de dibujarse). Con *GL_BLEND*, el color de las mezclas debe definirse con una llamada a la función, siendo el segundo parámetro *GL_TEXTURE_ENV_COLOR*. Conviene indicar que si usamos *GL_DECAL*, los efectos de iluminación no afectarán a los polígonos texturizados.

MULTIMAPA

En secuencias animadas, interesa usar texturas multimapa. Podemos tener texturas detalladas para los primeros planos, y más simples para los objetos lejanos. Si nos acercamos a un objeto, *OpenGL* va cambiado la textura para mejorar el detalle. El inconveniente, es que éstas tardan más en cargarse, aunque se reduce el tiempo de cálculo, al ahorramos el filtrado de aumento o reducción.

Sin texturas multimapa, *OpenGl* tiene que aplicar filtros lineales (los que pro

orcionan mayor calidad) para adaptar el amaño de la textura al del polígono, lo ue supone tiempo de cálculo desperdiiado. Si usamos texturas multimapa, al cercarnos al polígono, vamos cambiano las texturas por otras de mayor resoıción, y nos ahorramos filtrados sin ıermar la calidad de la imagen.

Las texturas multimapa se definen on el parámetro *level* de las funciones *lTexImage1D y glTexImage2D*, que inica la calidad de cada imagen. El nivel es el de máxima resolución. Una uena norma, es reducir el tamaño a la ıitad por cada nivel. Tenemos varias pciones para decidir qué nivel de texıra utilizar, según el tamaño del políono (según el filtro de minimización legido):

- *GL_NEAREST_MIPMAP_NEAREST*: usa la imagen más cercana a la resolución del polígono (la mas parecida en tamaño). Aplicaremos el filtro *GL_NEAREST* con estos mapas en caso de que, por ejemplo, la textura más pequeña sea más grande que el tamaño actual del polígono.
- *GL_NEAREST_MIPMAP_LINEAR*: usa la imagen más cercana a la resolución del polígono. Si es necesario, el filtro será el *GL_LINEAR*.
- *GL_LINEAR_MIPMAP_NEAREST*: interpola entre las dos imágenes más cercanas a la resolución del polígono. Filtro de tipo *GL_NEAREST*.
- *GL_LINEAR_MIPMAP_LINEAR*: interpola entre las dos imágenes más cercanas a la resolución del polígono. Usaremos el filtro *GL_LINEAR*.

Por supuesto, la última opción es la e mayor calidad, y la que más recursos onsume.

Con la librería *glu*, se pueden generar las texturas intermedias a partir de ı de alta resolución. Para ello, tenemos ıs funciones *gluBuild1dMaps* y *gluuild2dmaps*, que sustituyen a *gltexima-e1d* y *gltextimage2d*. El formato de esıs funciones es el siguiente:

```
uBuild1DMipmaps(target: TGLEnum; components,
    width: TGLint; format, atype: TGLEnum; data:
    Pointer)
```

```
uBuild2DMipmaps(target: TGLEnum; components,
    width, height: TGLint; format, atype: TGLE-
    num; Data: Pointer)
```

Si los resultados no son de nuestro grado, siempre podemos generar las exturas a mano.

EJEMPLO 2: Creación de listas de visualización para objetos y texturas.

```
function CreaObjeto:Cardinal;
const
      MatEsp : array [0..3] of Single = ( 1, 1, 1, 0);
var
  Quadric: PGLUquadricObj;
begin
  // Pedimos un identificador de lista libre...
  Result := glGenLists(1);
  glNewList(Result, GL_COMPILE);
   Quadric := gluNewQuadric;
     glColor3f(1, 1, 1);
     glColorMaterial(GL_FRONT, GL_DIFFUSE);
     glMaterialfv(GL_FRONT, GL_SPECULAR, @MatEsp);
     glMateriali (GL_FRONT, GL_SHININESS, 16);
     gluQuadricTexture(Quadric, GL_TRUE);
     gluQuadricDrawStyle(Quadric, GLU_FILL);
     gluQuadricOrientation(Quadric, GLU_OUTSIDE);
     gluSphere(Quadric, 5, 30, 30);
  gluDeleteQuadric(Quadric);
  glEndList;
end;

function CreaTexturas:Cardinal;
begin
  Result := glGenLists(1);
  glNewList(Result, GL_COMPILE);
  glTexImage2D(GL_TEXTURE_2D, 0, 3, 256, 256, 0, GL_RGB,
    GL_UNSIGNED_BYTE, Puntero);
  glEndList;
end;

function CreaTexturas2:Cardinal;
begin
  Result := glGenLists(1);
  glNewList(Result, GL_COMPILE);
  glTexImage2D(GL_TEXTURE_2D, 0, 3, 256, 256, 0, GL_RGB,
    GL_UNSIGNED_BYTE, Puntero2);
  glEndList;
end;
```

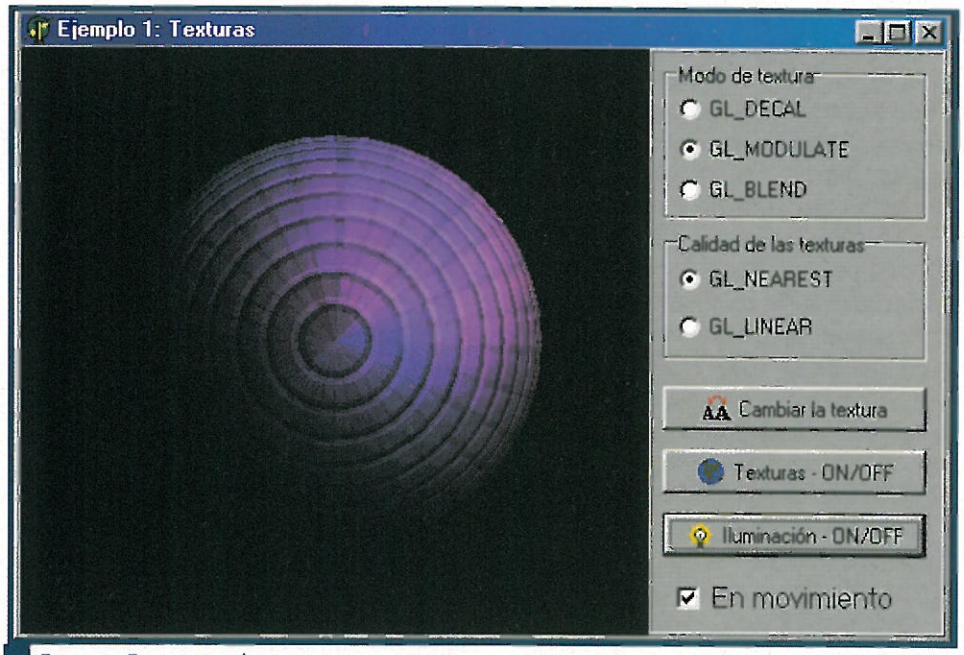


Figura 5. - Texturas con iluminación.

EJEMPLO 3: Código del evento onPaint del formulario 1.

```
//giramos la imagen
glRotatef(angulo,  1, 0, 1);
glRotatef(angulo*2, 0, 1, 0);

//guardamos la posicion inicial 3 veces
glPushMatrix;
glPushMatrix;
glPushMatrix;
//dibujamos el objeto con textura
glCallList(Textura2);
 glTranslatef(10,0,0);
 glCallList(Objeto);
//restauramos posicion inicial
//y volvemos a dibujar
glCallList(Textura);
 glPopMatrix;
 glTranslatef(-10,0,0);
 glCallList(Objeto);
glCallList(Textura2);
 glPopMatrix;
 glTranslatef(0,10,0);
 glCallList(Objeto);
glCallList(Textura);
 glPopMatrix;
 glTranslatef(0,-10,0);
 glCallList(Objeto);
```

LAS TEXTURAS EN LA PRÁCTICA

Hemos desarrollado dos ejemplos, en el primero veremos las distintas capacidades de *OpenGL* en tiempo real, como calidad de la textura, modo de colocarla, cómo afecta la luz a los objetos con texturas, etcétera. En el segundo veremos cómo podemos trabajar con listas de visualización, ya que desde la versión *1.1*, existe optimación también para almacenar las texturas en el *hardware* de texturas de nuestra tarjeta de video. Para el primero, vamos a partir del ejemplo 1 de la pasada entrega (esfera iluminada). Hemos hecho una serie de cambios: añadimos *gluQuadricTexture (Quadric, GL_TRUE)* para modificar el objeto con la representación de las texturas. Después activamos el sistema de texturas, mediante *glEnable (GL_TEXTURE_2D)*. Con esto, tenemos que elegir el sistema para combinar las texturas con los polígonos: *glTexEnvi (GL_TEXTURE_ENV, GL_TEXTURE_ENV_MODE,GL_MODULATE)*, es decir, los colores de la textura se mezclan con los del objeto.

Además, hay que definir qué va a hacer cuando la textura sea más pequeña que el objeto, para lo que hemos elegido la repetición, y lo activaremos así: *glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_REPEAT)*, y *glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_REPEAT)*.

Por último, indicamos el tipo de filtrado para reducir y ampliar la textura a la hora de la aplicación de ésta sobre un objeto: *glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR)* para reducir, y *glTexParameteri (GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR)* para aumentar. Se puede cambiar las características en tiempo de ejecución, para comprobar las diferencias entre unas y otras más fácilmente. Hay que advertir que *GL_LINEAR*, por el gasto computacional del filtrado lineal por píxel representado, puede bajar excesivamente el rendimiento general de la aplicación, sobre todo si deseamos un número de imágenes por segundo elevado.

Normalmente, con *GL_NEAREST* se consiguen unos resultados aceptables. De todas formas, siempre nos queda la solución de muchas aplicaciones (juegos sobre todo): uso por defecto del mejor filtrado, y cuando detectan movimiento, activan el de mayor rendimiento, pues con las imágenes en movimiento, es más difícil apreciar la calidad.

Por último, se establece la textura activa. Hasta que no la cambiemos, todo los objetos que se dibujen serán afectados por ella: *glTexImage2D(GL_TEXTURE_2D, 0, 3, Bitmap.Width, Bitmap.Height, 0, GL_RGB, GL_UNSIGNED_BYTE, datos)*, donde el parámetro *datos* debe ser el puntero que almacena el mapa de bits que anteriormente debemos haber cargado y convertido a *RGB*. Hay que invertir las componentes roja y azul de las texturas con la función *ArreglaImagen*. El programa carga las texturas necesarias (en formato *BMP*) y verifica que éstas sean cuadradas, de 24 *bits* de color, y con un ancho potencia de 2.

OpenGl selecciona automáticamente la mejor textura

En el segundo ejemplo, se utilizan las listas de visualización para optimar el trabajo. El uso de esta técnica aprovecha completamente nuestro *hardware*, y podréis comprobar fácilmente que la diferencia ahora si salta a la vista.

Los objetos y texturas se han construido con listas de visualización. En el evento *OnPaint* del formulario se hace lo siguiente (Listado 3): Se elige la textura deseada, y llamamos a su lista de visualización; Se elige el objeto, y llamamos a su lista de visualización. El resultado se aprecia en la Figura 9. El aumento de velocidad es considerable.

Quizá este sea uno de los apartado más sencillos de todo el curso. La facilidad con la que *OpenGL* trabaja con texturas, es simplemente inmejorable.

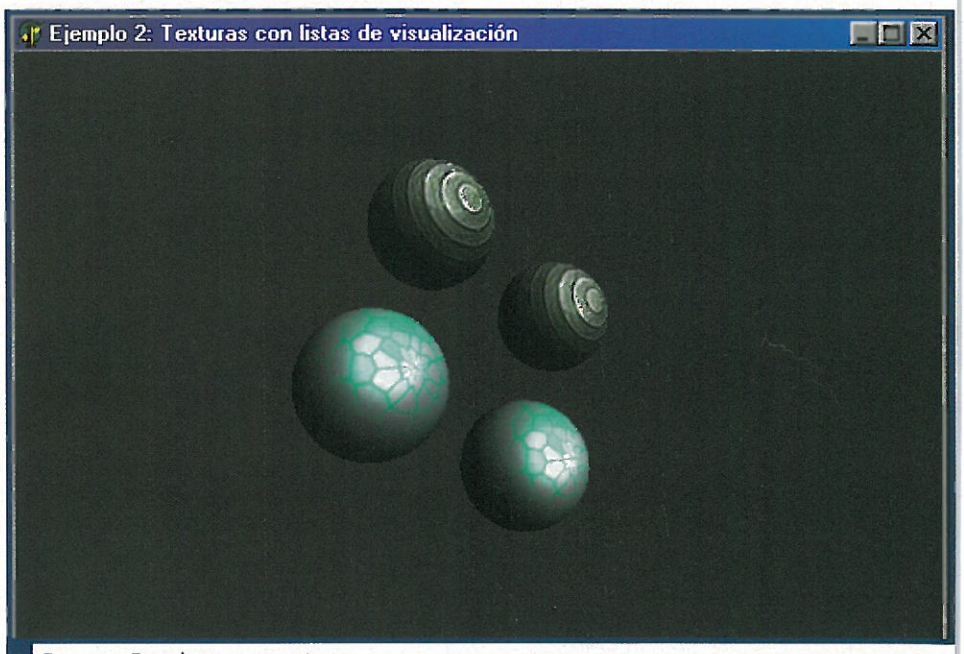


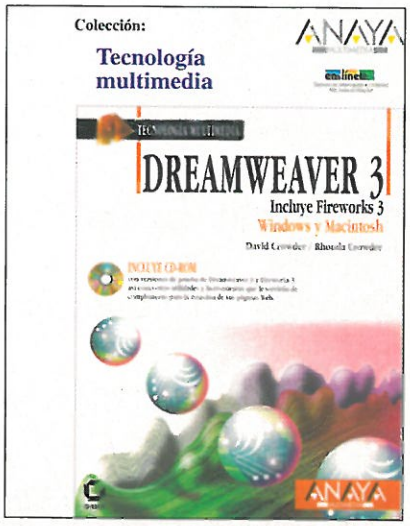
Figura 6. - Ejemplo 2 en ejecución.

DREAMWEAVER 3

Este manual está dedicado a uno de los programas de diseño de páginas *web* más completos y capaces del mercado, trabajando en combinación con su complemento perfecto, *Fireworks 3*. Ambos programas están preparados para trabajar en los sistemas *Windows* y *Macintosh,* y vienen dotados de un conjunto de utilidades y herramientas que

El lector encontrará en estas páginas, una explicación clara y sintética de los fundamentos de la creación de páginas *web*, acompañando a cada capítulo un ejemplo y ejercicio guiado de casos prácticos. Se pretende familiarizar al usuario con el uso de *Dramweaver,* para ir avanzando en el diseño de las páginas: añadir imágenes, vínculos y *URLs*, uso de color, tablas y formularios, marcos, bibliotecas, hojas de estilo en cascada, capas, *JavaScript*, *fireworks*, *lienzos* y *caminos*, etcétera.

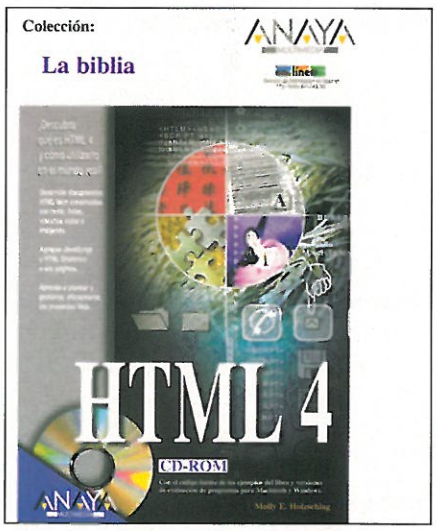
El libro está complementado con un *CD-ROM* con versiones de prueba de *Dremweaver 3* y *Fireworks 3* y con diversos ejemplos de páginas *web* e imágenes en color que ayudarán al lector en la confección de sus propias páginas, siguiendo los pasos de cada ejercicio guiado.



Editorial: Anaya Multimedia
Nº de páginas: 928
Nivel: Avanzado

Autor: David y Rhonda Crowder
Idioma: Español
Precio: 6.995Ptas. (I.V.A inc.)

HTML 4



El ritmo de desarrollo de las nuevas tecnologías relacionadas con *Internet*, obliga a los profesionales un constante proceso de reciclado y actualización de sus conocimientos.

Este libro pretende hacer llegar a los usuarios en general y a los profesionales los fundamentos y avances del *Hypertext Markup Language, HTML,* un sistema que permite integrar en un solo documento datos e información heterogéneos y de origen distinto, y con un nivel de dificultad y aprendizaje asequible a personas sin grandes conocimientos de protocolos y redes.

Desde los fundamentos y elementos básicos, hasta los objetos incrustados y la propia publicación, *HTML 4* es una obra de referencia para los profesionales que necesitan aclarar conceptos y actualizar técnicas de la publicación en *web*, o para los aficionados que desean aumentar sus conocimientos sobre esta industria en constante desarrollo.

El libro se acompaña de un *CD-ROM* con los códigos fuente de los ejemplos y diversas versiones de prueba y evaluación para *Windows* y *Macintosh*

Editorial: Anaya Multimedia
Nº de páginas: 992
Nivel: Avanzado

Autor: Molly Holzschlag
Idioma: Español
Precio: 6.995 Ptas. (I.V.A inc.)

Dudas técnicas

Cada vez nos cuesta más seleccionar las consultas entre tantas peticiones, a cual más interesante. La tuya puede ser la próxima. No dudes en mandarnos tu pregunta a la dirección solop@revistasprofesionales.com. En SÓLO PROGRAMADORES estamos encantados de atenderos.

PREGUNTA

Soy un asiduo lector de Sólo Programadores, mi nombre es Christian López Pasaron y les escribo de Argentina. Tengo un problema, que espero me lo solucionen como suelen hacerlo, a través de la revista.

En el número 62, la nota sobre *ASP* explica **cómo abrir archivos con el FileSystemObject,** pero en nuestro servidor no funciona. El error que me da es:

```
Microsoft JScript runtime error
    '800a0035'

File not found/pru5.asp, line 12
```

Le comento que tanto el código, como el archivo *listados.txt* están en la carpeta raíz del dominio. El servidor que utilizo es (según su página de *variables_del_servidor .asp*) es *Microsoft-IIS/4.0*.

Según la empresa que me provee el servicio, está todo bien, y me dicen que revise el código.

Ahhh..., otra cosa: para hacer referencia a un directorio... ¿cómo debo hacer? (poner /, //, \, \\, etc.) .

Muchas gracias.

RESPUESTA

Si observamos el tipo de error que nos remites, da la impresión de que el fichero que estás tratando de abrir no existe en el servidor. Si no quieres tener errores imprevistos antes de hacer cualquier operación tienes que estar muy seguro de que los ficheros se encuentran en el sitio adecuado, así como de que los permisos son los correctos.

El código que abre un fichero y lo lee, línea a línea, podría ser algo como lo que sigue:

```
<%
    var fso, f, r, s="";
    var ForReading = 1;

fso = new
    ActiveXObject("Scripting.FileSys-
    temObject");
    f =
    fso.OpenTextFile("C:\\TEXTO.TXT",
    ForReading);
    while (!f.AtEndOfStream) {
      s += f.ReadLine( ) + "<BR>";
```

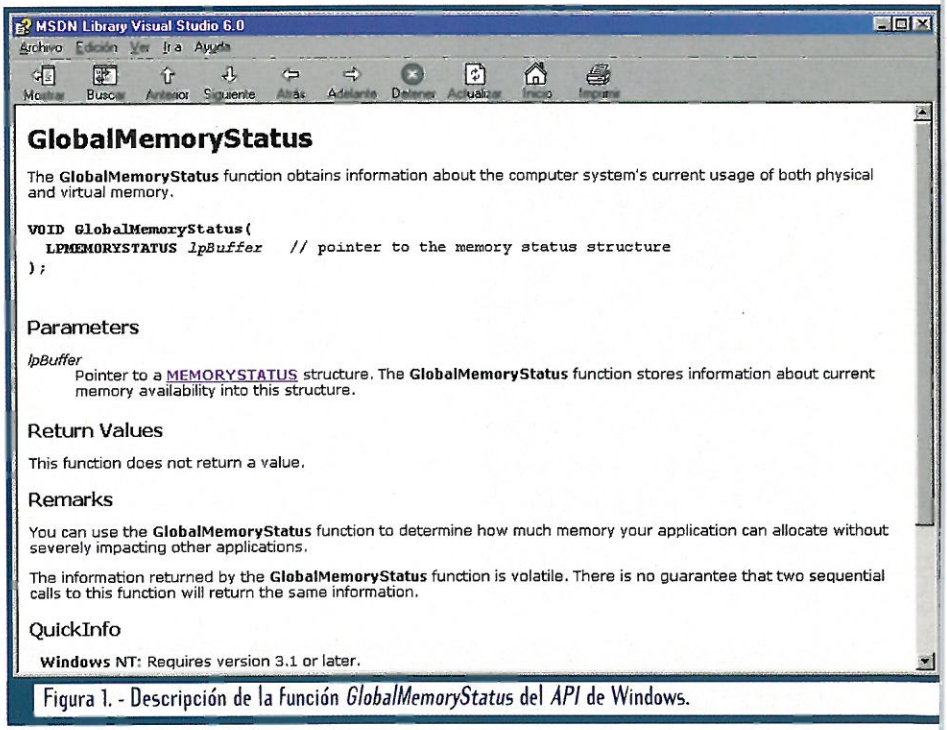


Figura 1. - Descripción de la función *GlobalMemoryStatus* del *API* de Windows.

```
}
    f.Close();
    %>
<%= s %>
```

Cuando se abre un fichero desde una página *ASP*, es necesario indicar al método *OpentextFile* una referencia absoluta. Las referencias a los directorios dependen del sistema operativo utilizado.

Teniendo en cuenta que se trata de páginas *ASP*, lo más seguro es que utilices *Windows*, ya que el carácter que se usa para separar directorios en una ruta absoluta es la barra "gallega" (\). En el ejemplo que hemos expuesto observarás que se utiliza doble barra.

Esta es la razón de que la barra "gallega" sea también el símbolo que aprovecha *Jscript* para los caracteres especiales, como el salto de línea, que es \n. De ahí que cuando se utiliza como un carácter más, debe ser "escapado", y esto se hace duplicándolo.

PREGUNTA

Hola compañeros, **me gustaría hacerme un mini programa** en *VisualBasic* que **me diera la memoria libre del equipo** en tiempo real y que se muestre junto al reloj de la barra de tareas. ¿Cuáles son las instrucciones que realizan estas dos operaciones?

Gracias.

Rubén Igual.

RESPUESTA

El programa del que nos hablas tiene dos funcionalidades distintas, que vamos a tratar por separado. Por un lado, tenemos que averiguar el estado de la memoria del ordenador, y por otro, tenemos que ser capaces de realizar una aplicación que se ejecute en el área de iconos de notificación. Como podrás suponer, directamente con *Visual Basic* no podemos implementar ninguna de estas dos características, por lo que será necesario acudir a llamadas del *API* de *Windows*.

La función *GlobalMemoryStatus* del *API* de Windows devuelve la información concerniente al estado actual del uso de la memoria del *PC*. Dicha información se proporciona a través de la estructura de datos *MEMORYSTATUS*. En la sección general de declaraciones de nuestra aplicación Visual Basic tenemos que añadir el siguiente código:

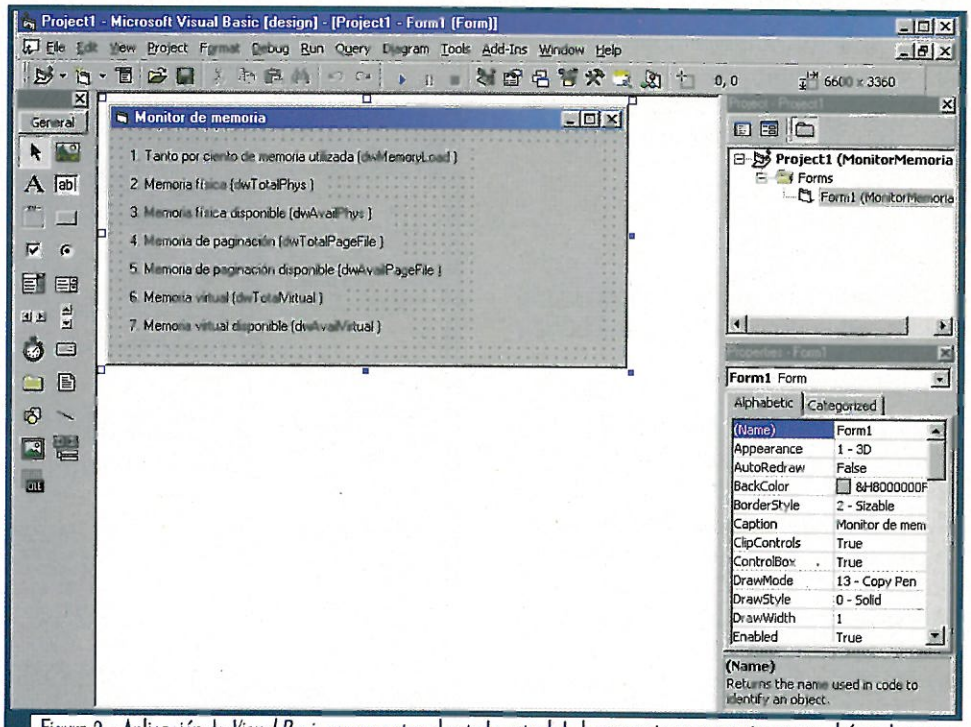


Figura 2. - Aplicación de *Visual Basic* que muestra el estado actual de la memoria y que se ejecuta en el área de iconos de notificación.

```
Private Declare Sub GlobalMemoryStatus
    Lib "kernel32" (lpBuffer As
    MEMORYSTATUS)
Private Type MEMORYSTATUS
    dwLength As Long
    dwMemoryLoad As Long
    dwTotalPhys As Long
    dwAvailPhys As Long
    dwTotalPageFile As Long
    dwAvailPageFile As Long
    dwTotalVirtual As Long
    dwAvailVirtual As Long
End Type
```

Los datos proporcionados por la estructura de datos *MEMORYSTATUS* son los siguientes:

- *dwMemoryLoad*
  Tanto por ciento de memoria que está siendo utilizada
- *dwTotalPhys*
  Número total de *bytes* de memoria física.
- *dwAvailPhys*
  Número total de *bytes* de memoria física disponible.
- *dwTotalPageFile*
  Número total de *bytes* de memoria de paginación.
- dwAvailPageFile
  Número total de *bytes* de memoria de paginación disponible.
- *dwTotalVirtual*
  Número total de *bytes* de memoria virtual.
- *dwAvailVirtual*
  Número total de *bytes* de memoria virtual disponible.

En el evento *Load* del formulario principal de la aplicación tenemos que insertar el código que lea el estado de la memoria en ese momento.

```
Dim MS As MEMORYSTATUS

MS.dwLength = Len(MS)
GlobalMemoryStatus MS
l1.Caption = MS.dwMemoryLoad & " %"
l2.Caption = Format(MS.dwTotalPhys /
    1024, fmt) & " Kb"
l3.Caption = Format(MS.dwAvailPhys /
    1024, fmt) & " Kb"
l4.Caption = Format(MS.dwTotalPageFile
    / 1024, fmt) & " Kb"
l5.Caption = Format(MS.dwAvailPageFile
    / 1024, fmt) & " Kb"
l6.Caption = Format(MS.dwTotalVirtual
    / 1024, fmt) & " Kb"
l7.Caption = Format(MS.dwAvailVirtual
    / 1024, fmt) & " Kb"
```

Las variable *l1···l7* son etiquetas que mostrarán la información devuelta

por la función *GlobalMemoryStatus*. Los valores numéricos devueltos por *dwTotalPhys*, *dwAvailPhys*, etcétera, han sido formateados utilizando la cadena:

```
Const fmt As String =
    "###,###,###,###"
```

De esta manera se muestran lo mejor posible después de su traducción a *Kbytes*.

Para hacer que una aplicación ponga su icono en el área de iconos de notificación es necesario también utilizar el API de Windows. La función *Shell_NotifyIcon* se encargará de hacer llegar a al sistema operativo las intrucciones precisas que hacen que éste muestre el icono, o lo quite, o cambie la cadena de texto que muestra cuando el ratón pasa por encima, etcétera. En la sección general de declaraciones de nuestra aplicación *Visual Basic* tenemos que añadir el siguiente código:

```
Private Type NOTIFYICONDATA
    cbSize As Long
    hWnd As Long
    uId As Long
    uFlags As Long
    ucallbackMessage As Long
    hIcon As Long
    szTip As String * 64
End Type

Private Const NIM_ADD = &H0
Private Const NIM_MODIFY = &H1
Private Const NIM_DELETE = &H2
Private Const NIF_MESSAGE = &H1
Private Const NIF_ICON = &H2
Private Const NIF_TIP = &H4

Private Const WM_MOUSEMOVE = &H200
Private Const WM_LBUTTONDOWN = &H201
Private Const WM_LBUTTONUP = &H202
Private Const WM_LBUTTONDBLCLK = &H203
Private Const WM_RBUTTONDOWN = &H204
Private Const WM_RBUTTONUP = &H205
Private Const WM_RBUTTONDBLCLK = &H206

Private Declare Function Shell_Notif-
    yIcon Lib "shell32" Alias
    "Shell_NotifyIconA" (ByVal dwMes-
    sage As Long, pnid As NOTIFYICON-
    DATA) As Boolean

Dim nid As NOTIFYICONDATA
```

Shell_NotifyIcon recibe dos parámetros: el mensaje con las instrucciones para el sistema, y una estructura de datos que se denomina *NOTIFYICONDATA*.

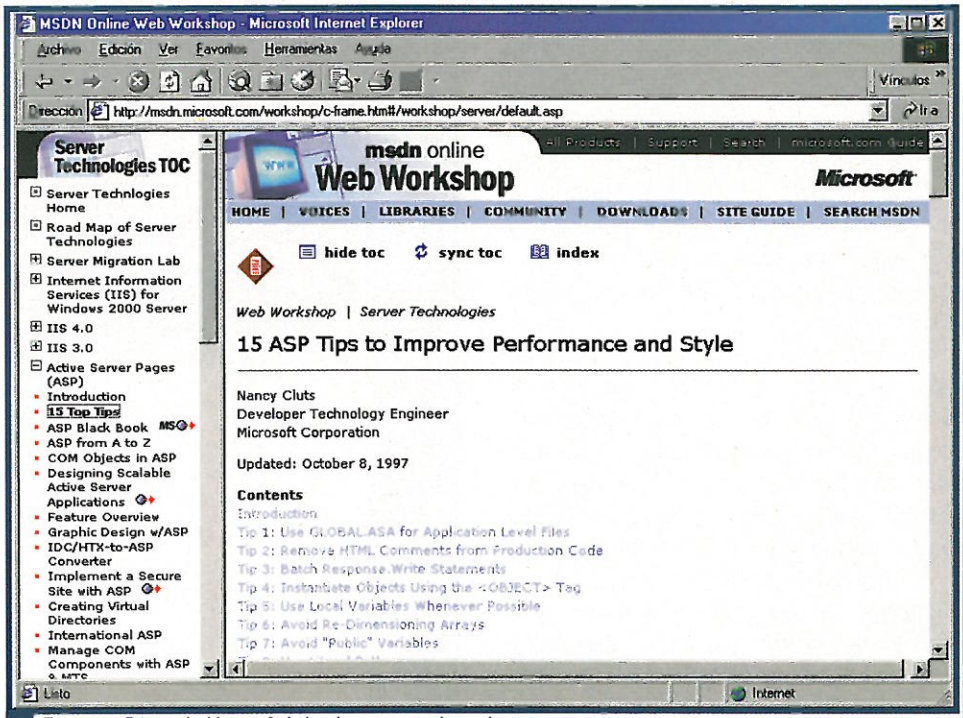


Figura 3. - Página de *Microsoft* dedicada a mejoras de rendimiento.

En el evento *Load* del formulario principal de la aplicación debemos ejecutar el siguiente código.

```
nid.cbSize = Len(nid)
nid.hWnd = Me.hWnd
nid.uId = vbNull
nid.uFlags = NIF_ICON Or NIF_TIP Or
    NIF_MESSAGE
nid.ucallbackMessage = WM_MOUSEMOVE
nid.hIcon = Me.Icon
nid.szTip = "Memoria libre" & vbNull-
    Char
Shell_NotifyIcon NIM_ADD, nid
Me.Hide
App.TaskVisible = False
```

Lo anterior hará que el formulario principal no se muestre. Windows mostrará el icono de la aplicación en el área de iconos de notificación. Si pasamos el ratón por encima se mostrará el mensaje "*Memoria libre*".

El resto de los eventos de la aplicación ha de programarse como sigue:

```
Private Sub Form_MouseMove(Button As
    Integer, Shift As Integer, X As
    Single, Y As Single)
    Dim msg As Long
    If Me.ScaleMode = vbPixels Then
        msg = X
    Else
        msg = X / Screen.TwipsPerPi-
    xelX
    End If
    Select Case msg
        Case WM_LBUTTONUP
            Me.WindowState = vbNormal
            Me.Show
        Case WM_RBUTTONUP
            Me.PopupMenu Menu
    End Select
End Sub

Private Sub Form_Resize()
    If Me.WindowState = vbMinimized
    Then Me.Hide
End Sub

Private Sub Form_Unload(Cancel As
    Integer)
    nid.cbSize = Len(nid)
    nid.hWnd = Me.hWnd
    nid.uId = vbNull
    Shell_NotifyIcon NIM_DELETE, nid
End Sub
```

Se muestra un menú cuando el usuario pulsa sobre el icono con el botón derecho del ratón. Este menú puede tener las opciones que deseemos. Cuando el usuario pulsa sobre el icono con el botón izquierdo del ratón, el formulario principal de la aplicación se muestra de manera normal. Entonces, si minimizamos la ventana principal de la aplicación, ésta "regresa" al área de iconos de notificación.

Consultoría • Mantenimiento • Distribución • Formación • Programación a medida • Certificación • Material didáctico • Cursos para empresas

Universe Linux™

La alternativa que su empresa estaba esperando

Linux está en boca de todos: potencia, versatilidad, seguridad con mayúsculas, economía. Y todos se hacen las mismas preguntas. ¿Debo cambiar a **Linux**? ¿Es cierto que **Linux** es más fiable en la empresa que Windows NT, 98 o Windows 2000? ¿Es cierto que **Linux** puede convivir con Windows intercambiando información de forma transparente? ¿Puedo ahorrar con soluciones basadas en **Linux** hasta el 60% del precio en sus programas equivalentes de Microsoft? ¿Por qué más del 65% de los servidores de Internet corren bajo **Linux**? ¿Quién me dará el soporte y la garantía necesarios?

La respuesta a todas las preguntas se llama ***Universe Linux***

Universe Linux es la primera empresa de nuestro país dedicada exclusivamente a ofrecer soluciones basadas en **Linux** para usuarios y empresas: distribuciones **Linux**, utilidades, consultoría, instalaciones a medida, mantenimiento, formación... Todo lo que Vd. necesite y al mejor precio del mercado.

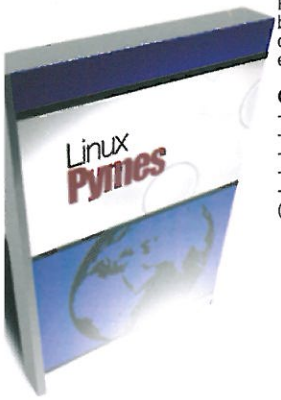
Puede que aún no sepa lo que ***Universe Linux*** puede hacer por Vd. Le invitamos a conocer nuestros productos con una demostración gratuita en nuestra sucursal de Madrid. Llámenos para concertar la demostración. Infórmese sin compromiso llamando al **91-356 69 08** o visitando nuestra página web:

http://www.u-linux.com

Llame hoy mismo. *Por fin* en LINUX hay alguien que RESPONDE.

Linux PYMES

La distribución pensada para empresas
Pvp. recomendado 9.900 pts.



Por fin, la primera distribución creada para las PYMES: robusta, fiable, sencilla de instalar. Incluye: StarOffice, escritorio KDE, manual de instalación paso a paso y, como oferta de lanzamiento,GRATIS, el programa LINUX FAX SERVER, valorado en 14.900 pts. (*)

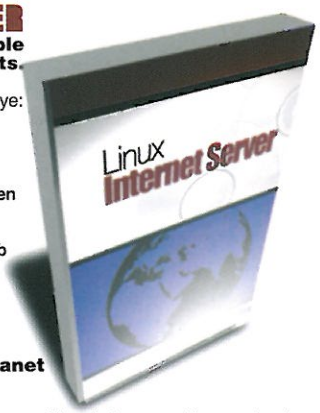
Características:
- Instalación en modo gráfico con pantallas de ayuda
- Configuración X completa antes de la instalación de paquetes
- Flexibilidad en los modos de instalación Server y Workstation
- Utilidades de automontaje
- XFree 3.3.5-3, Kernel 2.2.12-20, KDE 1.1.2, GTK, etc.

(*) Oferta válida hasta el 1 de abril del 2000.

FORMACIÓN
Cursos de Linux todos los niveles,
Básico
Administrativo
Seminarios

Linux INTERNET SERVER

El servidor más fiable
Pvp. recomendado 49.950 pts.



El servidor de Internet pensado para su empresa incluye:
***Servidor de correo electrónico, páginas Web y FTP.**
***Servidor de proxy y caché.**
***Gestión y mantenimiento de listas de correo.**
***Alojamiento del Dominio y Dirección IP fija.**
***Gestión de usuarios y autentificación de los mismos en la entrada al sistema.**
***Control y visualización de usuarios conectados.**
***10 niveles de seguridad en el servidor de páginas Web local.**
***Programación de eventos de conexión con Internet.**
Ver demostración en http:// www.u-linux.com/demo

Linux FAX SERVER

Ahorre enviando todos sus fax por la intranet
Pvp. recomendado 24.900 pts.

Se acabaron las colas para enviar un fax. El primer servidor de fax para Linux selecciona entre los tramos horarios más económicos de su operador de comunicaciones y se adapta a ellos, economizando en cada envío. Evita desplazamientos dentro de la propia empresa ahorrando tiempo y molestias al utilizar la intranet para acceder al servidor. Crea las estadísticas necesarias para analizar los gastos por cada usuario. Economía y comodidad garantizadas en sus comunicaciones.

LINUX WATCHDOG CONTROLLER

El guardián de su empresa en Internet
Pvp. recomendado 69.950 pts. (*)

El primer sistema de CONTROL TOTAL sobre Internet en su empresa. Limitación en el tiempo de conexión, listas de direcciones, chat, en general los puertos más representativos del sistema. Estadísticas de conexión por usuario, por sites accedidas, por distribución del tiempo según operador. Limitaciones individuales o por grupos de usuarios. El sistema corta la conexión cuando detecta que el usuario ha sobrepasado los límites de sus privilegios y le envía un mensaje al supervisor. LINUX WATCHDOG CONTROLLER es una herramienta imprescindible en la empresa moderna para el control del uso de Internet. El 90% de los empleados de las empresas con acceso a Internet desde su puesto de trabajo navegan con fines personales: chats, compras, etc. Esta herramienta erradica el problema realizando auditorías individualizadas de cada usuario, ya que almacena direcciones accedidas, tiempo de estancia, información recibida y enviada, etc. LINUX WATCHDOG CONTROLLER es una utilidad imprescindible para los Administradores del Sistema y Jefes de Personal. Pruébelo sin compromiso.

(*) Hasta 10 puestos de trabajo. Consultar según necesidades superiores.
Los precios no incluyen IVA.

Universe Linux *le brinda las mejores opciones para* **hacer negocio:**

Linux Training Center

Formación a todos los niveles

Si dispone de una academia de informática Vd. puede tener el LINUX TRAINING CENTER oficial de su población. Universe Linux formará a su personal y le proporcionará todo el material didáctico necesario y la certificación LINUX TRAINING CENTER. Además, le incluiremos en nuestra Lista de Centros Autorizados en todas las acciones publicitarias promocionando su negocio. (*)
La más amplia gama de cursos para todos los niveles y necesidades en formación: Introducción a Linux, Administración de Sistemas, Instalación y configuración de Servidores web, Ofimática en Linux, etc.
Sistema de franquicias con número limitado de licencias por provincia.

(*) Más de 250.000 impactos cada mes en prensa especializada y general.

Linux Solution Provider

Una apuesta segura para emprendedores. Servicios en Linux

Si es Vd. socio o propietario de una empresa de Servicios Informáticos puede obtener para su provincia la representación en *exclusiva* de los productos **Universe Linux** y la certificación **LINUX SOLUTION PROVIDER** para su empresa de servicios, distribuyendo, instalando y dando soporte a la más amplia gama de productos Linux, con los mejores márgenes comerciales del mercado. Además, contará con el asesoramiento de nuestros técnicos y ayuda on line.
Un negocio de rentabilidad asegurada, sin cánones de entrada, compatible con su actividad y clientes habituales. No deje pasar la oportunidad e infórmese en el **91-356 69 08.**

Linux Developer

La mejor opción para desarrolladores

Universe Linux ofrece a los profesionales independientes del mundo de la programación la posibilidad de adquirir todos los conocimientos para trabajar en el mundo Linux. Incluye el material didáctico y el software necesario a un precio excepcional. Sólo para desarrolladores.

Y también: STAROFFICE, APPLIXWARE, ANTIVIRUS, bases de datos relacionales, servidores de ficheros, y mucho más.

Distribuidores

Universe Linux S.L. España ofrece las mejores condiciones para distribuidores, con descuentos desde el 25% hasta el 40% según tramos de compra. Infórmese sin compromiso en el teléfono: (91) 356 69 08.
Si desea solicitar una demostración gratuita de alguno de los programas, póngase en contacto con nosotros.

UNIVERSE LINUX, LINUX SOLUTION PROVIDER, LINUX TRAINING CENTER, LINUX PYMES, LINUX INTERNET SERVER, LINUX WATCHDOG CONTROLER, son marcas registradas de U-LINUX, S.A. España.

WINDOWS NT, WINDOWS 98, WINDOWS 2000, MICROSOFT son marcas registradas de MICROSOFT CORPORATION.

se Linux S.L • C/ Conde de Vilches 21 • 28028 MADRID • Teléfono: (91) 356 69 08 • Fax: (91) 355 28 95 • http://www.u-linux.com

El Poder es Tuyo
¡Elige Ahora!

Special Program Available for Resellers

Disponible en:

abc analog, s.l.
✆ 91 634 20 00

GTI
✆ 91 660 08 30

redha

www.redhat.co
Red Hat (ESPAÑA) s